Unsupervised Methods for Learning and Using Semantics of Natural Language

Vom Fachbereich Informatik
der Technischen Universität Darmstadt
genehmigte

Dissertation

zur Erlangung des akademischen Grades
Doctor rerum naturalium
(Dr. rer. nat.)

vorgelegt von

Martin Riedl, M.Sc., Dipl. Inf. (FH)

geboren in Mannheim

Tag der Einreichung: 11. Januar 2016
Tag der Disputation: 24. Februar 2016

Referenten: Prof. Dr. rer. nat Chris Biemann, Darmstadt
Prof. Dr. Phil. Anders Søgaard, PhD, Copenhagen

Darmstadt 2016
D17

Dedicated to my grandfather Karl Gäßler

Abstract

Teaching the computer to understand language is the major goal in the field of natural language processing. In this thesis we introduce computational methods that aim to extract language structure — e.g. grammar, semantics or syntax — from text, which provides the computer with information in order to understand language.

During the last decades, scientific efforts and the increase of computational resources made it possible to come closer to the goal of understanding language. In order to extract language structure, many approaches train the computer on manually created resources. Most of these so-called supervised methods show high performance when applied to similar textual data. However, they perform inferior when operating on textual data, which are different to the one they are trained on. Whereas training the computer is essential to obtain reasonable structure from natural language, we want to avoid training the computer using manually created resources.

In this thesis, we present so-called unsupervised methods, which are suited to learn patterns in order to extract structure from textual data directly. These patterns are learned with methods that extract the semantics (meanings) of words and phrases. In comparison to manually built knowledge bases, unsupervised methods are more flexible: they can extract structure from text of different languages or text domains (e.g. finance or medical texts), without requiring manually annotated structure. However, learning structure from text often faces sparsity issues. The reason for these phenomena is that in language many words occur only few times. If a word is seen only few times no precise information can be extracted from the text it occurs. Whereas sparsity issues cannot be solved completely, information about most words can be gained by using large amounts of data.

In the first chapter, we briefly describe how computers can learn to understand language. Afterwards, we present the main contributions, list the publications this thesis is based on and give an overview of this thesis.

Chapter 2 introduces the terminology used in this thesis and gives a background about natural language processing. Then, we characterize the linguistic theory on how humans understand language. Afterwards, we show how the underlying linguistic intuition can be

operationalized for computers. Based on this operationalization, we introduce a formalism for representing words and their context. This formalism is used in the following chapters in order to compute similarities between words.

In Chapter 3 we give a brief description of methods in the field of computational semantics, which are targeted to compute similarities between words. All these methods have in common that they extract a contextual representation for a word that is generated from text. Then, this representation is used to compute similarities between words. In addition, we also present examples of the word similarities that are computed with these methods.

Segmenting text into its topically related units is intuitively performed by humans and helps to extract connections between words in text. We equip the computer with these abilities by introducing a text segmentation algorithm in Chapter 4. This algorithm is based on a statistical topic model, which learns to cluster words into topics solely on the basis of the text. Using the segmentation algorithm, we demonstrate the influence of the parameters provided by the topic model. In addition, our method yields state-of-the-art performances on two datasets.

In order to represent the meaning of words, we use context information (e.g. neighboring words), which is utilized to compute similarities. Whereas we described methods for word similarity computations in Chapter 3, we introduce a generic symbolic framework in Chapter 5. As we follow a symbolic approach, we do not represent words using dense numeric vectors but we use symbols (e.g. neighboring words or syntactic dependency parses) directly. Such a representation is readable for humans and is preferred in sensitive applications like the medical domain, where the reason for decisions needs to be provided. This framework enables the processing of arbitrarily large data. Furthermore, it is able to compute the most similar words for all words within a text collection resulting in a distributional thesaurus. We show the influence of various parameters deployed in our framework and examine the impact of different corpora used for computing similarities. Performing computations based on various contextual representations, we obtain the best results when using syntactic dependencies between words within sentences. However, these syntactic dependencies are predicted using a supervised dependency parser, which is trained on language-dependent and human-annotated resources.

To avoid such language-specific preprocessing for computing distributional thesauri, we investigate the replacement of language-dependent dependency parsers by language-independent unsupervised parsers in Chapter 6. Evaluating the syntactic dependencies from unsupervised and supervised parses against human-annotated resources reveals that the unsupervised methods are not capable to compete with the supervised ones. In this chapter we use the predicted structure of both types of parses as context representation in order to compute word similarities. Then, we evaluate the quality of the similarities, which provides an extrinsic evaluation setup for both unsupervised and supervised dependency parsers. In an evaluation on English text, similarities computed based on contextual representations generated with unsu-

pervised parsers do not outperform the similarities computed with the context representation extracted from supervised parsers. However, we observe the best results when applying context retrieved by the unsupervised parser for computing distributional thesauri on German language. Furthermore, we demonstrate that our framework is capable to combine different context representations, as we obtain the best performance with a combination of both flavors of syntactic dependencies for both languages.

Most languages are not composed of single-worded terms only, but also contain many multi-worded terms that form a unit, called multiword expressions. The identification of multiword expressions is particularly important for semantics, as e.g. the term *New York* has a different meaning than its single terms *New* or *York*. Whereas most research on semantics avoids handling these expressions, we target on the extraction of multiword expressions in Chapter 7. Most previously introduced methods rely on part-of-speech tags and apply a ranking function to rank term sequences according to their multiwordness. Here, we introduce a language-independent and knowledge-free ranking method that uses information from distributional thesauri. Performing evaluations on English and French textual data, our method achieves the best results in comparison to methods from the literature.

In Chapter 8 we apply information from distributional thesauri as features for various applications. First, we introduce a general setting for tackling the out-of-vocabulary problem. This problem describes the inferior performance of supervised methods according to words that are not contained in the training data. We alleviate this issue by replacing these unseen words with the most similar ones that are known, extracted from a distributional thesaurus. Using a supervised part-of-speech tagging method, we show substantial improvements in the classification performance for out-of-vocabulary words based on German and English textual data. The second application introduces a system for replacing words within a sentence with a word of the same meaning. For this application, the information from a distributional thesaurus provides the highest-scoring features. In the last application, we introduce an algorithm that is capable to detect the different meanings of a word and groups them into coarse-grained categories, called supersenses. Generating features by means of supersenses and distributional thesauri yields an performance increase when plugged into a supervised system that recognized named entities (e.g. names, organizations or locations).

Further directions for using distributional thesauri are presented in Chapter 9. First, we lay out a method, which is capable of incorporating background information (e.g. source of the text collection or sense information) into a distributional thesaurus. Furthermore, we describe an approach on building thesauri for different text domains (e.g. medical or finance domain) and how they can be combined to have a high coverage of domain-specific knowledge as well as a broad background for the open domain. In the last section we characterize yet another method, suited to enrich existing knowledge bases. All three directions might be further extensions, which induce further structure based on textual data.

The last chapter gives a summary of this work: we demonstrate that without language-dependent knowledge, a computer can learn to extract useful structure from text by using computational semantics. Due to the unsupervised nature of the introduced methods, we are able to extract new structure from raw textual data. This is important especially for languages, for which less manually created resources are available as well as for special domains e.g. medical or finance. We have demonstrated that our methods achieve state-of-the-art performance. Furthermore, we have proven their impact by applying the extracted structure in three natural language processing tasks. We have also applied the methods to different languages and large amounts of data. Thus, we have not proposed methods, which are suited for extracting structure for a single language, but methods that are capable to explore structure for "language" in general.

Zusammenfassung

Welche Schritte sind notwendig, damit ein Computer Sprache versteht? Dies ist die fundamentale Frage, mit der sich der Bereich der Computerlinguistik beschäftigt. In den letzten Jahrzehnten haben überwachte Systeme beachtliche Erfolge erzielt. Diese Systeme lernen Sprachstruktur (z. B. Grammatik, Semantik oder Syntax) anhand im Text manuell markierten (annotierten) Sprachstrukturen und sind anschließend in Lage diese in unbekannten Texten vorherzusagen. Die Generierung solcher Daten ist allerdings zeitaufwendig. Weiterhin sind die meisten überwachten Systeme nur in der Lage gute Vorhersagen auf Texten zu treffen, die den manuell annotierten Texten ähnlich sind. Um einem Computer generelles Wissen über Sprachen beizubringen sind daher unüberwachte Methoden von Interesse, die Sprachstrukturen anhand von Textkollektionen selbst extrahieren. Dies ermöglicht eine Adaption solcher Methoden sowohl an verschiedenartigen Texten (medizinische Texte, Zeitungstexte) als auch an andere Sprachen. Um dabei Sprachstrukturen zuverlässig zu lernen, ist es notwendig, dieses Wissen aus großen Textkollektionen zu extrahieren. Im Rahmen dieser Arbeit präsentieren wir unüberwachte Methoden, welche die Bedeutung von Wörtern — sprich ihre Semantik — erfassen, und zeigen deren Leistungsfähigkeit.

Im ersten Kapitel geben wir eine Einführung zu automatischen Verfahren, die Computern ein gewisses „Verständnis" von Sprache ermöglichen. Anschließend beschreiben wir den wissenschaftlichen Beitrag dieser Arbeit und nennen die Publikationen, auf denen diese Arbeit aufbaut.

Die Grundlagen sowie die Terminologie, die in dieser Arbeit verwendet werden, sind Bestandteil des zweiten Kapitels. Danach beschreiben wir aus linguistischer Sicht wie Menschen Sprache verstehen und stellen eine Verbindung zu den Themen in dieser Arbeit her. Weiterhin definieren wir eine formale graphenbasierte Repräsentation, die im Rahmen dieser Arbeit verwendet wird. Anschließend zeigen wir, wie diese Repräsentation eingesetzt werden kann um Wörter und ihre Kontextrepräsentation (z. B. benachbarte Wörter) darzustellen.

Das Kapitel 3 beschreibt existierende Methoden zur Berechnung von semantischen Ähnlichkeiten zwischen Wörtern, die allesamt auf der distributionellen Hypothese basieren. Diese besagt, dass sich Wörter ähnlicher sind je häufiger sie im gleichen Kontext auftreten. Als Kon-

text werden dabei z. B. benachbarte Wörter betrachtet oder auch der Satz in dem ein Wort auftritt. Wir erklären die Funktionsweise dieser Methoden und zeigen Beispiele für Wort- und Dokumentenähnlichkeiten.

Da ein Wort nicht nur eine Bedeutung haben kann, ist der Kontext, in dem Wörter verwendet werden, wichtig. Anhand des Kontextes in dem ein Wort vorkommt, können Menschen oftmals bereits dessen Bedeutung erkennen. In Kapitel 4 beschreiben wir eine Methode, die — mittels eines statistischen Modells zur Themenerkennung — Texte in thematisch kohärente Abschnitte aufteilt. Wir demonstrieren die Leistungsfähigkeit unseres Systems mittels zweier Datensätze und erzielen auf diesen bessere oder vergleichbare Resultate zu aktuellen Forschungsergebnissen.

Im Kapitel 5 präsentieren wir eine Methode, mit der Ähnlichkeiten zwischen Wörtern berechnet werden können. Unsere Methode verfolgt dafür einen „symbolischen Ansatz", der Wörter mit einer Kontextrepräsentation darstellt, welche für Menschen nachvollziehbar ist. Durch den Vergleich dieser Kontextrepräsentationen zwischen zwei Wörtern können Wortähnlichkeiten berechnet werden. Unsere Methode kann auf beliebig große Textkollektionen angewendet werden und ermöglicht die Berechnung von Ähnlichkeiten für alle Wörter im Text. Dies führt zu einem sogenannten distributionellen Thesaurus. Aufgrund des symbolischen Ansatzes können dadurch Begründungen für die Ähnlichkeit zweier Begriffe gegeben werden, was im Rahmen von sensitiven Anwendungen (z. B. medizinische Anwendungen) von Bedeutung ist. In der Folge zeigen wir den Einfluss verschiedener Parameter und vergleichen unseren Ansatz mit Standardmethoden. Die besten Ergebnisse werden dabei von unserer Methode unter der Verwendung syntaktischer Kontextmerkmale erzielt. Diese werden allerdings mit einem überwachten Dependenzparser generiert, der mit Hilfe von manuell erstellten Trainingsdaten trainiert wird.

Aus diesem Grund ersetzten wir in Kapitel 6 den Kontext von überwachten Dependenzparsern durch Kontext von unüberwachten Dependenzparsern zur Berechnung von Wortähnlichkeiten. Dieses Experiment soll die Leistungsfähigkeit von unüberwachten Dependenzparsern zeigen, wenn diese als Kontextmerkmal eingesetzt werden. Für die Berechnung distributioneller Thesauri auf englischsprachigen Texten können keine Verbesserungen gegenüber überwachten Dependenzparsern erzielt werden. Allerdings erzielen wir bessere Resultate unter Verwendung eines unüberwachten Dependenzparsers bei deutschen Texten. Weiterhin ist unser Verfahren zur Ähnlichkeitsberechnung aus Kapitel 5 in der Lage verschiedene Kontextmerkmale sowohl von unüberwachten als auch überwachten Dependenzparsern zu kombinieren und erzielt mit dieser Kombination die besten Ergebnisse.

Ein weiterer Schritt um Sprache zu verstehen, ist zu erkennen, welche Wörter eine Einheit bilden. Dazu wird in Kapitel 7 eine Methode vorgestellt, die sogenannte Mehrwortbegriffe (z. B. New York) extrahiert und dazu Informationen aus einem distributionellen Thesaurus verwendet. Im Vergleich zu existierenden Verfahren benötigt unsere Methode kein sprachabhängiges Wissen. Die Leistungsfähigkeit unseres Ansatzes wird dabei anhand von französischen und

englischen Texten evaluiert. Im Vergleich zu Methoden aus der Literatur erzielt unser Ansatz die besten Ergebnisse.

Im Kapitel 8 demonstrieren wir wie Sprachstrukturen, die von unseren Methoden extrahiert werden, in Anwendungen integriert werden können. Zuerst stellen wir einen generellen Ansatz vor, mit dem unbekannte Wörter, die nicht im Training von überwachten Methoden enthalten sind, durch Wörter aus den Trainingsdaten ersetzt werden können. Wird stellen dazu ein Verfahren vor, welches dafür einen distributionellen Thesaurus verwendet, der auf einer großen Textkollektion berechnet wird. Das Verfahren sucht dazu die unbekannte Wörter in dem berechneten Thesaurus nach und ersetzt diese durch ein ähnliches Wort das in den Trainingsdaten enthalten ist. Wir testen dieses Verfahren anhand einer Methode, die Wortarten bestimmt. Durch die Verwendung der „Ersetzungen“ können Verbesserungen erzielt werden. Als Zweites zeigen wir den Einfluss von semantischen Merkmalen in einem System, das Wörter innerhalb eines Satzes durch Wörter mit einer gleichen Bedeutung ersetzt. Dabei zeigt sich, dass die semantischen Merkmale aus einem distributionellen Thesaurus die größte positive Auswirkung auf die Performanz des Systems haben. Aufgrund der Mehrdeutigkeit von Sprache können Wörter nicht nur eine Bedeutung haben sondern mehrere. Um die verschiedenen Bedeutungen von Wörtern zu erkennen, stellen wird im letzten Anwendungsfall eine Methode vor, die Wörter aufgrund ihrer Bedeutungen in grobe Kategorien einteilt. Sowohl diese Kategorien also auch Informationen aus einem distributionellen Thesaurus werden anschließend in einem System zur Erkennung von Namensentitäten (z. B. Orte, Namen oder Unternehmen) evaluiert. Erneut werden Verbesserungen unter Verwendung dieser Merkmale beobachtet.

In Kapitel 9 beschreiben wir zukünftige Forschungsideen. Zuerst zeigen wir wie semantische Ähnlichkeiten aus Kapitel 5 dazu verwendet werden können zusätzliche Informationen in Wörtern zu speichern. Dies ermöglicht z. B. die Domäne der Textkollektion zu erfassen oder Wortbedeutungen direkt zu unterscheiden. Anschließend stellen wir ein automatisches Verfahren vor, mit dem Thesauri für spezielle Textarten (z. B. medizinische Texte) erstellt werden können. Das nächste Verfahren beschreibt, wie Methoden aus dieser Arbeit dazu verwendet werden können um existierende Wissensdatenbanken, wie Taxonomien oder Ontologien, anzureichern.

Das letzte Kapitel fasst die Forschungsinhalte dieser Arbeit zusammen: unter Verwendung großer Datenmengen können wir zeigen, dass unsere Methoden in der Lage sind, Sprachstrukturen sprachunabhängig zu extrahieren. Dabei haben wir Lösungen für drei Forschungsprobleme dargelegt: die thematischen Textsegmentierung, die Berechnung von Wortähnlichkeiten und die Erkennung von Mehrwortbegriffen. Weiterhin haben wir die Leistungsfähigkeit unserer Modelle in drei Anwendungen demonstriert. Dies ist ein weiterer Schritt dem Computer beizubringen, Sprache nicht nur anhand von Beispielen zu lernen, sondern selbstständig Strukturen aus Text zu extrahieren.

Acknowledgements

Writing this thesis would have been impossible without the help and support of many people. I want to thank all of them and want to apologize if anyone has been omitted.

First of all I want to thank my supervisor Chris Biemann for his endless support and advices. Every time I got stuck in some research direction he had some ideas how to break free. He did not only taught me about computational linguistics, which I did not know beforehand, but definitely pitched me to tackle tasks with unsupervised methods. During that time Chris was not only my supervisor but also became a friend. I also want to thank Anders Søgaard not only for reviewing this thesis but also for providing inspirations and ideas, which influenced my work.

In particular I want to thank all my colleagues at Language Technology and UKP for their support and my student assistants. I also want to thank Beáta Megyesi who did not only review some chapters of this thesis but advised me to re-write and re-structure parts of this thesis. Whereas these advices have been laborious, they made this thesis a much better one. A huge thanks goes to all my further proofreaders who did an incredible job in helping me to improve my writing skills: Héctor Martínez Alonso, Tobias Baum, Darina Benikova, Or Biran, Johannes Daxenberger, Emma Franklin, Mike Kestemont, Barbara Plank, Simone Paolo Ponzetto, Stephen Roller, Eugen Ruppert, Alexander Panchenko, Jörg Schönfisch and Cäcilia Zirn.

Without my parents and my family I would not be the person I am. I want to thank especially my parents for their never-ending support and for never having any doubts about the next steps I took. Furthermore, I want to thank all my friends, especially Daniel Sellmann, Markus Sperl, my brother-in-beer Stefan Pfenning and Sven Molzen, who always showed me that there is a life aside of research.

Without my love Katharina I might have gone crazy. I want to thank her for spending all the years with me and encouraging me when I got stuck in work. She never complained about me sometimes being a workaholic and traveling around the world to join conferences.

This work has partly been supported by the Hessian research excellence program "Landes-Offensive zur Entwicklung Wissenschaftlich-Ökonomischer Exzellenz" (LOEWE).

Contents

List of Abbreviations

ABL	Alignment Based Learning
AHMM	Aspect Hidden Markov Model
AP	average precision
BHT	Big Huge Thesaurus
BNC	British National Corpus
CCG	combinatory categorial grammar
CCL	common cover links
CCM	Constituent-Context Model
CL	Computational Linguistics
CRF	Conditional Random Fields
CUI	concept unique identifier
CV	cross-validation
CW	Chinese Whispers
DMV	Dependency Model with Valence
DP	Dynamic Programming
DS	distributional semantics
DT	distributional thesaurus
ERC	Est Républican Corpus
ESG	English Slot Grammar
EM	expectation–maximization
GAP	generalized average precision
HDP	Hierarchical Dirichlet Process
HMM	Hidden Markov Model
IDC	International Data Consortium
IQR	Interquartile Range
IR	information retrieval

LCC	Leipzig Corpora Collection
LDA	Latent Dirichlet Allocation
LL	Log-Likelihood
LMI	Lexicographer's Mutual Information
LSA	Latent Semantic Analysis
MAP	mean average precision
GWMT	Grady Wards Moby Thesaurus
MT	Machine Translation
MWE	multiword expression
MWT	Merriam-Webster Webster Collegiate Thesaurus
MWU	Multi Word Unit
NED	Neutral Edge Direction
NER	Named Entity Recognition
NLP	Natural Language Processing
NYT	New York Times
OOV	out-of-vocabulary
pLSA	probabilistic Latent Semantic Analysis
PMI	Pointwise Mutual Information
POS	part of speech
PTB	Penn Treebank
RT	Roget's Thesaurus 1911
RNN	recurrent neural network
SD	Structure Discovery
SVD	singular value decomposition
TDT	Topic Detection Track
TM	Topic Model
TS	Text Segmentation
TT	TextTiling
UDP	Unsupervised Dependency Parser
UMLS	Unified Medical Language System
UP	unsupervised parser
VSM	vector space model
WD	WindowDiff
WSD	Word Sense Disambiguation
WSI	Word Sense Induction
WSJ	Wall Street Journal

Chapter 1

Introduction

According to Georgiev (2006), teaching the computer to understand natural language is the "ultimative goal" in computational linguistics. In order to understand natural language, the computer needs to know the semantics of language, which is the meaning of the word or phrase in context. Here the context is provided by the phrase, clause, sentence or text the word occurs. In this thesis, we focus on the research question on how to develop methods, which help the computer to understand natural language by learning from text directly. This can be achieved by extracting structure — e.g. syntactic, grammatical or semantic structure — from text.

Language is the fundamental tool of communication for humans. Whereas most people know how to operate a computer, they often fail at communicating with computers. This comes due to the limitations of computers to understand the needs of humans, which is a problem of language mismatches. Computers are designed to understand the language of commands consisting of ones and zeros. Such a language differs immensely from the language humans use.

If computers are able to understand language, they can extract relevant information from text automatically. When humans need to extract relevant information from text, they need to read the text, which is time-consuming. In contrast, a computer equipped with language learning and understanding skills, can process text much faster and provide information about the text. Humans already use computers to search information with search engines. However, these engines do not fully understand language and mostly answer search queries with a list of somehow matching webpages. Then, again humans need to manually find the answer by reading the results.

The research field, which deals with computational methods to understand language, can be divided into two fields: Computational Linguistics (CL) and Natural Language Processing (NLP). CL focuses on the research of how linguistic knowledge is retrieved in order to understand and use language. Often these approaches rely on linguistic theories. According to

Manning and Schütze (1999) this field is dominated by the *rationalist* concept, which assumes that "*knowledge in the human mind is [...] fixed in advance*". In contrast, NLP usually does not rely on linguistic theories but concentrates on the processing of language. This is nowadays mainly performed based on statistical methods, which aim to extract language structure in order to solve engineering problems. With the term language structure we refer to any morphological, syntactical or semantic information that is explicitly and implicitly contained in language. In NLP computational methods use this structure in order to improve natural language processing systems, such as e.g. search engines or machine translation, rather than validating linguistic theories. Nevertheless, a strict separation of both fields is impossible, as nowadays both fields influence each other.

At the beginning of the development of both fields, mainly *rule-based systems* have been developed and have been applied to unstructured data — data that is not formalized in a predefined format and consists mostly of text — in order to extract structure. To extract structure from text, specialists generated rules and patterns. These rule-based systems have the advantage of being simple and representing human reasoning. Additionally, reasoning for the extracted structure can be given by the responsible rules.

Whereas each rule on its own can be simple, the interaction and relations among several rules are not trivial. Because of this, it is challenging to add new rules to existing systems. Additionally, such systems are static and are limited to extract only structure, for which the rules are designed. Thus, these systems are incapable of learning new structure. Furthermore, rule-based systems heavily rely on humans, who need to refine, modify or add rules, in order to allow these systems to achieve good performances when the data they are applied changes.

A better way to detect structure found in natural language can be achieved with statistical models. In the 90s, many rule-based systems have been replaced by *supervised machine learning* approaches. These approaches rely on data, which is manually annotated with the structure of interest. Then, statistical learning algorithms are applied to learn the annotated structure with the objective to predict this structure based on previously unseen text. Whereas these methods convince with good performances on previously unseen data, the main issue is the requirement of annotated data.

The annotation of linguistic structure in text is a complex task, which is time-consuming and requires, depending on the structure, experts to obtain reliable annotations. In order to address human errors and the various options for structural annotation, several experts commonly annotate text. This makes the annotation process not only time-consuming but also expensive. After the annotation of text, statistics need to be derived to obtain the agreement between the annotators and to decide which annotations are used for the training. In addition to annotated data, many *supervised learning* approaches use lexical resources (e.g. dictionaries or thesauri) to yield better results. Many lexical resources are, similar to the annotated data, manually created. Although considerable efforts have been made in order to generate both lexical resources and annotated data for the English language, the coverage of these resources

is marginal for many other languages. As language changes over time, resources additionally need to be updated. Furthermore, within one language there are varieties, as different vocabularies are used in different working areas (e.g. finance or medical) or also among different media (e.g. newspaper, Web text or Twitter), which are called domains. Additionally, in different domains words can have different meanings.

Therefore there is a need for methods, which do not require annotated data but acquire structure from text directly. Such methods are called *unsupervised methods*. Most unsupervised methods perform a clustering of structure, which is driven by a text collection, called corpus, without any annotations. This enables the adaptation of such methods to various languages and to different text domains. However, most unsupervised methods have the disadvantage that in comparison to their supervised counterpart, they have a lower performance. Often the structure, extracted with unsupervised methods, varies slightly from the one extracted by supervised methods. This often makes the evaluation of the structure of unsupervised methods more difficult, when using existing annotated data, as this data is often not perfectly suited for the evaluation.

In this thesis we introduce methods that learn from textual data directly and do not require language-dependent processing. We expect that one method is not able to extract all structure at once, which is needed to understand language. Thus, we require methods, which enrich text with structure and use this structure to gain further structure. This can be accomplished with the paradigm of *Structure Discovery (SD)* described by Biemann (2011).[1] This

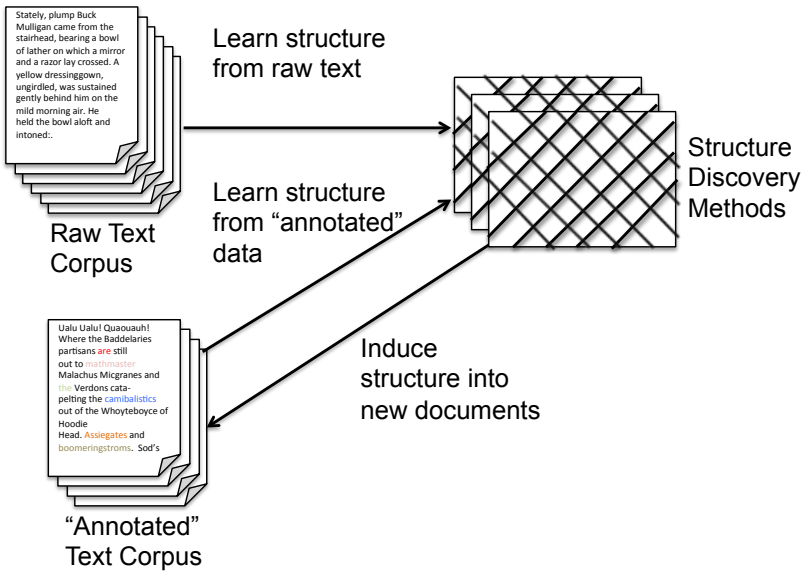


Figure 1.1: Graphical illustration[o of the Structure Discovery (SD) paradigm.

paradigm (see Figure 1.1) relies on methods, which are capable to learn structure from raw

[1]Whereas the SD paradigm was first defined by Biemann (2011), such iterative approaches have already been used before e.g. in Clark (2000); Klein and Manning (2004).

text. Learning directly from text can be achieved with e.g. unsupervised learning methods. The learned structure is then induced back into the text. This can be the same text used for training or previously unseen text. In the next step, structure extraction can be performed either with supervised or unsupervised algorithms. The advance of this paradigm is that the extracted structure enriches the further learning steps and gives additional improvements in a self-learning fashion.

In this thesis, we concentrate on the development of methods, which examine the semantics of words and phrases, learned from text. For this, we introduce unsupervised methods, which can be applied knowledge-freely and language-independently. Such methods can be used to extract structure from text and can follow the SD paradigm in order to extract further structure. Furthermore, we aim to design our methods in such a way that they can be applied to arbitrarily large data.

Here, we described how computers learn to discover structure from text. In Section 2.3, we establish the connection between such a computational approach to the linguistic theory on how humans understand natural language.

1.1 Main Contributions

In the following we describe steps towards teaching the computer to understand natural language that are introduced in this thesis.

When computers get in touch with human language in textual form, they consider text as a sequence of characters without any meaning. The segmentation of words and sentences based on this character sequence is a task, which is rather simple for Western languages that use whitespaces and punctuation marks.[2] When humans read text, they are also capable to detect topic changes. Thus, we first introduce a robust algorithm, which is able to detect topically coherent segments in text by using a statistical semantic model. Evaluating this method against two datasets demonstrates the performance of the method. This provides also an extrinsic testbed for the evaluation of the parameters of the statistical semantic model.

One of the main information for understanding text is the semantics of words and sentences, which can be achieved by representing words with the context (e.g. neighboring words, sentences or documents) they appear, which forms the context representation. In this thesis we consider a symbolic representation, which represents a word not by dense numeric vectors but uses a meaningful representation (called symbolic representation). As previous work has shown that using large amounts of text results to better representations, we require the method to be scalable to arbitrarily large amounts of textual data. We also allow the context representation to use language-dependent knowledge in order to find the best context representation. Using such semantic representations allows the computation of word similarities.

[2]For several languages no whitespaces are used to separate words (e.g. Chinese, Japanese, Thai and Vietnamese). For these languages the segmentation is still an open problem, which is not targeted within this thesis.

In order to compute similarities between all words, resulting to a distributional thesaurus, methods are required, which can perform similarity computations between all words in an efficient manner.

We expect to obtain the best performing context representation for words by using information generated by supervised methods. Thus, we also investigate the impact of unsupervised and language independent methods, which generate context representations that are similar to the language-dependent one. Furthermore, this allows testing, whether the method is able to combine context representations from various sources.

In many languages, a unit-forming term is often not only composed of a single word, but also of multiple words, called multiword expressions. Thus, a computer also needs to detect multiword expressions in order to understand that e.g. *hot dog* is not a *dog* that is *hot* but something to eat. This shall be accomplished with an unsupervised method, which ranks sequences of succeeding words according to their multiwordness.

Furthermore, words tend to comprise not only of a single meaning, but often have several meanings. For example the word *jaguar* can both be the car brand and the animal. To fully understand the meaning of a word in its context, it is important to detect the different meanings of a word. This constitutes the next challenge, which we tackle using semantic similarities.

Whereas these contributions constitute a step towards an unsupervised language understanding for computers, information from the extracted structure can also be plugged into supervised methods used in NLP. This is demonstrated for three applications that use information from the extracted structure: part-of-speech (POS) tagging, lexical substitution and Named Entity Recognition (NER).

For all research questions, except the last, we set the constraints that we want to gain the information in an unsupervised fashion using as little linguistic preprocessing as possible. We mainly target on the English language as for this language the most resources exist, which enable an appropriate evaluation of our approaches.

1.2 Publication Record

We have published some of the main contributions of this thesis in peer-reviewed conferences, workshop and journals of major events such as ACL, NAACL, EMNLP and COLING. The chapters that are build upon these publications are marked accordingly. A full publication list can be found in Appendix B.

Martin Riedl, Chris Biemann (2015): 'A Single Word is not Enough: Ranking Multiword Expressions Using Distributional Semantics', in: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, pp. 2430–2440, Lisboa, Portugal (Chapter 7).

Martin Riedl, Michael Glass, Alfio Gliozzo (2014): 'Lexical Substitution for the Medical Domain', in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, pp. 610–614, Doha, Qatar (Section 8.2)

Martin Riedl, Irina Alles, Chris Biemann (2014): 'Combining Supervised and Unsupervised Parsing for Distributional Similarity', in: Proceedings of the 25th International Conference on Computational Linguistics, COLING 2014, pp. 1435–1446, Dublin, Ireland (Chapter 6)

Martin Riedl, Richard Steuer, Chris Biemann (2014): 'Distributed Distributional Similarities of Google Books over Centuries', in: Proceedings of the Fourth International Conference on Language Resources and Evaluation, LREC 2014, pp. 1401–1405, Reykjavik, Iceland (Chapter 5)

Martin Riedl, Chris Biemann (2013): 'Scaling to Large^3 Data: An efficient and effective method to compute Distributional Thesauri', in: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, pp. 884–890, Seattle, WA, USA (Chapter 5)

Chris Biemann, **Martin Riedl** (2013): 'Text: Now in 2D! A Framework for Lexical Expansion with Contextual Similarity', Journal of Language Modelling, JLM, 1(1), pp. 55–95 (Chapter 5 and Section 8.1)

Martin Riedl, Chris Biemann (2012): 'Text Segmentation with Topic Models', Journal for Language Technology and Computational Linguistics, JLCL, Vol. 27, No. 1, pp. 47–70 (Chapter 4)

Martin Riedl, Chris Biemann (2012): 'How Text Segmentation Algorithms Gain from Topic Models', in: Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2012, pp. 553–557, Montreal, Canada (Chapter 4)

Martin Riedl, Chris Biemann (2012): 'TopicTiling: A Text Segmentation Algorithm based on LDA', in: Proceedings of the Student Research Workshop of the 50th Meeting of the Association for Computational Linguistics, ACL 2012, pp. 37–42, Jeju, Republic of Korea (Chapter 4)

Martin Riedl, Chris Biemann (2012): 'Sweeping through the Topic Space: Bad luck? Roll again!', in: Proceedings of the Joint Workshop on Unsupervised and Semi-Supervised Learning in NLP held in conjunction with EACL 2012, ROBUS-UNSUP 2012, pp. 19–27, Avignon, France (Chapter 4)

1.3 Outline

Before we present approaches to tackle the research questions, we introduce the terminology used within this thesis in **Chapter 2**. Then, we give a brief overview of the linguistic theories, which our methods are built on. Additionally, this chapter covers brief descriptions about processing text in NLP and introduces a formalization in order to represent the context of words.

In **Chapter 3** we give a broad overview of existing computational semantic models, which can be used to compute semantic similarities between documents and words. We demonstrate the methods with examples and present their characteristics.

We the task of text segmentation in **Chapter 4** and compare an algorithm using words with an algorithm using topical information from a statistical method. Furthermore, we introduce a method for text segmentation that uses statistical topical information of words. Based on this method, we examine the influence of parameters used to compute the statistical topic information. In addition, we show the performance of this method in contrast to recent methods based on two datasets.

Chapter 5 introduces a symbolic framework for computing similarities between words, resulting to a distributional thesaurus. This framework aims at performing the computation on arbitrarily large amounts of textual data. We examine the influence of various parameters, different-sized text collections and different context representations. Additionally, we compare our approach to standard and recent methods.

Some previously used context representations used for similarity computation require language-dependent syntax information retrieved using supervised syntactic parsers. In **Chapter 6** we replace supervised syntactic parsers with unsupervised syntactic parsers and compare the quality of similarities when using them to generate context representations. Furthermore, we show the performance when combining the context representation from both parser types.

Language is not only composed of single words, but also of multiword expressions such as *hot dog* or *New York*. In **Chapter 7** we propose a method for retrieving multiword expressions from raw text using a distributional thesaurus that was computed following the approach introduced in Chapter 5. The performance of our method is demonstrated on English and French and compared to standard approaches.

In the previous chapters we introduced methods for extracting structure from raw text. We examine the benefits of such structure in **Chapter 8** when using information from this structure in supervised NLP methods. When applying supervised methods to unseen text, they often show inferior performance when observing words that have been seen during training. Thus, we introduce a general method that uses a distributional thesaurus to replace unknown words with the most similar words that are contained in the training data. We show the performance of this method based on the detection of part of speech (POS) tags. The second task, called lexical substitution, is about replacing words in context with words of the same

meaning. We show improvements by incorporating information from distributional thesauri (DTs) as features into the system. In the third section we introduce a method, which is able to disambiguate the different meanings of words. Based on a system for the detection of named entities (e.g. organizations, names or locations) in text, we demonstrate the impact of this information.

We describe future work based on our methods in **Chapter 9** and summarize the findings of this thesis in **Chapter 10**. Additionally, we highlight the contributions of this work, describe the limitations of the methods and exemplify which structure can be induced in text with our methods.

Chapter 2

Background

> We demand rigidly defined areas of doubt and uncertainty!
>
> *Douglas Adams, in The Hitchhiker's Guide to the Galaxy*

For the uninitiated, we start with definitions and explanations about terminology used within this thesis, including standard preprocessing steps in Natural Language Processing (NLP). The second section gives a brief introduction into machine learning, and commonly used measures to evaluate the performance of machine learning algorithms. Then we describe how humans understand language from a linguistic perspective. In Section 2.4, we introduce a general method for transforming text into *language elements* and its contextual representation. This transformation is used both in Chapters 3 and 5. In the following section, we introduce graph-based definitions for describing how we can operationalize the contextual representation defined in the previous section. Afterwards, we present weightings commonly applied to *language elements* and its contextual representation. The chapter concludes with a description of measures to compute the similarities between *language elements*.

2.1 Definitions

2.1.1 Wording of Words And Beyond

There is no a-priori clear definition of what constitutes a "word". This section thus defines how we define the notion. Whereas Crystal (2008) describes three different meanings for the term word, we define a word pragmatically.

We define a **word** as a sequence of characters (e.g. letters, numbers, punctuations) that builds a unit. In many languages, words can be detected within text as they are delimited by

space and inter-punctuation. If not otherwise stated a word is not comprised by more than one word. Within this thesis, we mostly use the word **term** to name single words or a sequence of words, which forms a unit. Examples for a term can be *dog* but also *hot dog*. In contrast to terms, a **multiword expression (MWE)** is always composed of more than one word.

When we process a sentence like "*He had always had a definition.*", we count seven **tokens**, as we consider punctuation marks as tokens. We define a **token** as the occurrence of a word within the text, which has a unique position, e.g., in the document. In contrast to tokens, a **type** is used to describe a term in general and can be e.g. a dictionary entry. In general it is used to count the total number of distinct words within a text. Using the example sentence above we observe six types as the word *had* occurs twice. In NLP, text is often decomposed into fragments of subsequent tokens, which are called **n-gram**. The n specifies the number of succeeding tokens. For example, the sentence above contains, among others, the following 3-grams: *He had always* and *had always had*. Some n-grams, which are commonly used, have its own name like the unigram ($n = 1$), the bigram ($n = 2$) and the trigram ($n = 3$). We define a **lexical item** as a unit, which can be e.g. a term, an MWE, a sentence or even a document but needs to be explicitly visible within the text. In contrast, a more general item is the **language element**, which can be a lexical item but without the restrictions of its visibility within text. Thus, a language element can also contain implicit information like part of speech (POS) (e.g. nouns) or a concatenation of a lemma and its POS (e.g. mouse_noun).

2.1.2 Grouping Words Together

Words can be grouped together by different granularities. One of them would be grouping them by their POS. A more coarse-grained separation is obtained by grouping words into the following categories: function and content words.

Function words have very little lexical meaning and mostly are relevant to express grammatical relations between words. These words are mostly pronouns, determiners and prepositions and constitute a limited set of words.

Content words are also called open class words as the number of terms is not limited and new content words can evolve. They are composed of nouns, verbs, adjectives and adverbs and are words, which have a lexical meaning. Content words can be also defined as all words, which are not function words.

Similar to function words **stopwords** are words with little lexical meaning. Normally stopwords are subsets of function words with high frequency. Lists of stopwords are often used in NLP tasks in order to remove words without a positive or negative influence of the meaning of e.g. a sentence. Typically used stopwords are articles (*the*, *a*), conjunctions (*and*, *or*), negations (*not*) and prepositions (e.g. *of* , *in* or *at*). Additionally, punctuation marks are often used as stopwords and are also called **stopsymbols**.

2.1.3 Processing of Words and Sentences

Words are comprised out of **morphemes**, which is the smallest meaningful unit of a language. The word *undefined* consists of the prefix *un*, the root word *defin* and the suffix *ed*.

Each word has a base form, called **lemma**. This instance is found within lexicons at the beginning of an entry. In text, tokens often occur in different variations, e.g. the lemma of the token *had* is *have*. Converting a token into a lemma is called lemmatization, which is a general preprocessing step in NLP. In order to apply lemmatization, the POS of a word needs to be known.

For this a **POS tagger** can be used to predict the POS of each word. POS taggers are algorithms, which are mostly trained on annotated data to induce a model. This model is then in turn used to predict POS tags. For example in Figure 2.1, for each token in the sentence the POS tags are shown above the token. The abbreviations used within the thesis are described in Table 2 in Marcus et al. (1994). For example, *DT* is used for determiners, *NN* for singular nouns, *VB* for verbs, *VBZ* for verbs in the third person singular and *JJ* is used for adjectives.

With so called **dependency parsers** the syntactical structure of a sentence is extracted. Whereas syntactic structure can be described in forms of trees (see p. 407ff in Manning and Schütze (1999)), we focus on linguistic structure in form of dependencies between words, called dependency grammar (see Figure 2.1). A dependency relation is drawn from a *governor* word

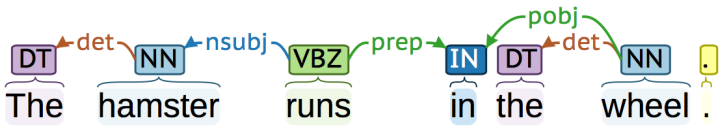


Figure 2.1: This figure shows a sentence, where each token is marked with its POS tags. Additionally, dependency relations between the terms are shown, which are predicted using the Stanford parser (de Marneffe et al., 2006). The visualization is based on BRAT (Stenetorp et al., 2012).

to a *dependent* word and is labeled with a dependency relation name. The *governor* word is the head of its *dependent* word. For example, the word "*hamster*" is the *governor* for the determiner word "*The*" but is governed itself by the verb "*runs*". In dependency grammars, each word has by definition only one *governor* word but can have several *dependent* words. The most important word within a sentence is the verb, which also *governs* most words. Thus, it is also annotated as the *root* of all structure, which is the only dependency that is not a relation between two tokens. Based on the example in Figure 2.1 this is the token "*runs*". As the root relation is rarely used in NLP, we do not show it in the figure. Each dependency relation is labeled with an abbreviated relation name: for example *nsubj* refers to a nominal subject relation, *prep* to a preposition relation and *pobj* denotes relation between an object and a preposition. A full listing of the dependency relations used by the Stanford parser is described in (de Marneffe and Manning, 2015).

Dependency relations are important for many NLP applications and are used as features in Chapters 5 and 8. According to the graph in Figure 2.1 we observe that the verb "*runs*" is directly connected to the noun "*hamster*" but a direct connection to the second noun "*wheel*" is missing. In the collapsed Stanford dependency scheme, direct links between content words are established by removing mostly preposition relations. In order to keep the information of the connections removed, the skipped word (e.g. proposition) is added to the relation name. As an example, the green relations `(prep; runs; in)` and `(pobj; wheel; in)` from Figure 2.1 are merged to the relation `(prep_in; runs; wheel)` as illustrated in Figure 2.2.

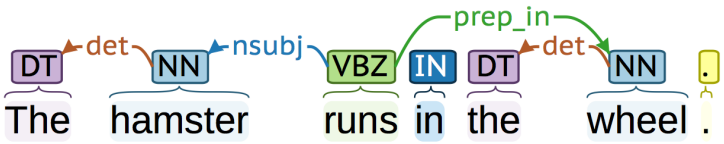


Figure 2.2: This graph shows the same example sentences as in Figure 2.1. In contrast, we show collapsed dependencies predicted by the Stanford parser. These dependencies add prepositions into the relation name and merge dependencies.

Using collapsed dependencies improves results when using them as contextual representation for computing similarities between terms as we have shown in (Ruppert et al., 2015b).

2.2 Primer on Machine Learning

The field of machine learning copes with algorithms, which automatically learn patterns from data in order to make predictions on unseen data. In this section, we briefly describe supervised, unsupervised and semi-supervised learning.[3]

2.2.1 Supervised Learning

In supervised learning, we have data comprised from *instances*, which are labeled *positive* and *negative*.[4] Such annotated data is also called *gold standard*. The goal is the learning of a function, called *classifier*, which is able to predict the correct label for a previously unseen instance. In order to achieve a general classifier the labeled data is split into *training data*, which is used to learn the classifier, and a *test set* that is not used during the training and serves to demonstrate the performance for previously unseen data. This prevents training a classifier, which performs perfectly on training data but is not general enough to perform well on previously unseen data. A perfect adaptation to the training data but a bad generalization is called *overfitting*. The performance of a classifier is highly dependent on the representation

[3] A deeper introduction into machine learning can be found in e.g. (Bishop, 2006; Hastie et al., 2009).

[4] Whereas there are also classifiers, which are able handle more than two classes, we mainly focus on binary classifier in this section. An overview of multi-label classifiers is given by Tsoumakas and Katakis (2007).

of an instance. These representations are called features. For example, training a classifier to predict POS tags for German, the capitalization of a word serves as a good feature for predicting nouns. Supervised learning algorithms are used in Sections 8.1, 8.2, 8.3.

2.2.2 Unsupervised Learning

Unsupervised machine learning targets finding hidden structure from *unlabeled data*, which is also called *knowledge discovery* (Murphy, 2012). In contrast to supervised learning, there is less specification about the information that is extracted, which makes the discovery of patterns more difficult. Additionally, there is no obvious error metric available due to the lack of labeled data.

The most commonly used unsupervised algorithms are clustering or dimensionality reduction. However, all algorithm, that do not require any learning are called unsupervised algorithms. As learning structure from unlabeled data is one of the major targets of this thesis, all further chapters focus on unsupervised learning methods.

2.2.3 Semi-Supervised Learning

Semi-supervised learning is a mixture between supervised and unsupervised learning. The goal of this learning strategy is to obtain a better classifier when training on both an unlabeled and a labeled dataset than using each one alone. The approach is motivated as labeled data is expensive to generate whereas unlabeled data is mostly available for free. Thus, there is normally less labeled data available, while the amount of unlabeled data is large. Within this thesis, we do not consider semi-supervised learning.

2.2.4 Splitting the Training Data

In most machine learning approaches the labeled data is split into three parts: training data, development data and test data. The majority of the data (e.g. 80%) is used for training the classifier. In order to tune for different features, the development set, also called dev-set, is used (e.g. 10%). The final performance is reported based on the test set (e.g. 10%) to avoid an overfitting and bias according to the development set.

Due to the efforts needed to obtain labeled data, the typically available amount of labeled training data is often too small to be split into three parts. To obtain reliable results with few labeled data the **cross-validation** is often used. Applying an N-fold cross-validation requires splitting the labeled data into N equally-sized folds. Then each of the N folds is used as test set. The remaining $N-1$ sets are merged and used for training the classifier. Based on the test set the performance can be measured. This process is performed N times.

For computing a final score we differentiate between micro- and macro-averaged scores. The macro-averaged score is computed by the performance for each fold separately and reports

on the average of all folds. Considering micro-averaged scores, we merge the classification results of all N test sets and then compute the score. In most evaluations N is set to 5 or 10. We report on results using a 10-fold cross-validation in Section 8.1. When setting N to the number of instances, we always test the performance based on one instance, which is called *leave-one-out cross-validation*. This is useful if the training data of e.g. 9 folds in a 10-fold cross-validation setting is too small. The measures to report on the performance of a classifier are explained in the next section.

2.2.5 Measuring the Performance

The performance of machine learning algorithms is reported mostly using **precision**, **recall**, **F-measure** and **accuracy**. When we apply a classifier to predict previously unseen data based on a binary decision (true and false), we observe four categories of instances as shown in Figure 2.3 and explained next:

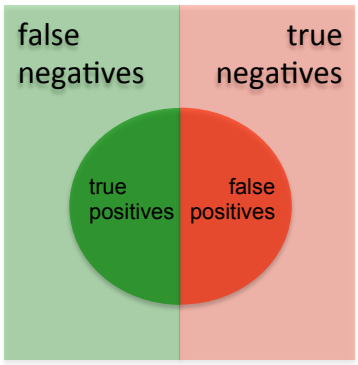


Figure 2.3: Based on the training data, we have true (false negatives and true positives) and false (true negatives and false positives) labeled instances. When applying a classifier to unseen data, some true instances are correctly classified as true (true positives) and other true instances are labeled as false incorrectly (false negatives). The same is obtained for false annotated instance resulting in false positive and true negative instances.

- true positives (tp): the number true instances that have correctly been classified true.
- false negatives (fn): the number of instances that are labeled true but have been classified as false.
- true negatives (tn): the number of false labeled instances that are classified correctly as false instances.
- true negatives (fn): the number of false labeled instances that are classified as true.

Based on these four categories the following four measures can be computed using the formulas shown below:

$$\text{precision} = \frac{tp}{tp + fp} \tag{2.1}$$

$$\text{recall} = \frac{tp}{tp + fn} \tag{2.2}$$

$$\text{F1-measure} = 2\frac{precision \cdot recall}{precision + recall} = \frac{2 \cdot fp}{2 \cdot tp + fp + fn} \tag{2.3}$$

$$\text{accuracy} = \frac{tp + tn}{fn + tp + tn + fp}. \tag{2.4}$$

The **precision** reports how many of the instances, which are classified as positive are also labeled true. The **recall** reports the number of true positives divided by all instances, which are labeled as true instances. Often precision and recall have an inverse correlation. The highest precision of 1.0 can e.g. be achieved by classifying one instance as true, which is labeled as true. Contrary, a perfect recall of 1.0 is obtained when classifying all instances as true ones. Thus, classifiers are often optimized according to the **F1-measure**, which is the harmonic mean between precision and recall. Another measure is the **accuracy**, which specifies the ratio of correctly classified instances.

In information retrieval, classifiers are used to rank results. Thus, the target is to obtain ranked results where the correct answers are ranked higher than the incorrect ones. For this reason, the precision is often performed for the top k-ranked results, which is abbreviated as $P@k$. As the highest ranked results can be considered as positive labeled instances, the so-called precision at k is the number of positive instances within the k results divided by k. We show results based on $P@k$ in Chapter 7 and gives examples of this measure in Section 5.4.1. Further evaluation measures for ranking based evaluations are presented in Chapter 7 and Sections 5.4.1 and 8.2.

2.3 Linguistic Theory on Language Understanding

According to Winograd (1972) humans understand a sentence when hearing or reading it by using their knowledge and intelligence. For the process of understanding language we do not rely only on speech and text. We agree with Vigliocco et al. (2014) who claim that human learning should be considered as a "multimodal phenomenon". Not only do we understand language from speech signals, but also from visual information e.g. gestures or mental images about objects that humans connect with a word. Whereas humans mostly learn language in an interactive manner, they also achieve a better understanding of language by reading written language. Especially for learning a new language, written language is an important source of information for humans who are able to read (Rosa and Eskenazi, 2011). However, this does

not apply to children and infants, who use different information and processes to learn and understand language, as described for example by Wells (1987) and Halliday (1993). In this thesis we focus on learning language structure from textual data.

According to Halliday (1993) human learning is a "process of making meaning — semiotic process". De Saussure initiated the field of semiotics and defined that language is made up of signs (de Saussure, 1959). Regarding to his semiotic theory, a sign is composed into a *signifier* (French: signifiér) and a *signified* (French: signifiant). Whereas de Saussure defines a *signifier* as a phonetic component, we adopt the definition by Palmer (1981) where a *signifier*, called symbol, is a *language element* e.g. word, lemma, sentence, paragraph, etc. The *signified*, also called concept, is the object that comes to our mind for a *signifier*. Thus, the alphabetic string "cat" is the symbol for the concept of cat. According to de Saussure the symbol (*signifier*) and the concept (*signified*) form the *sign*. Whereas the concept of a sign is always fixed and unique, it can have different symbols (*signifiers*). Symbols are mostly different among languages (e.g. the English word *cat* is different from the German word (*Katze*) for the same concept), but a single symbol can also refer to different concepts. A refinement of de Saussure's theory is the *semiotic triangle* introduced by Ogden and Richards (1989), as shown in Figure 2.4. The symbol is again the *language element* (e.g. word, sentence), the *referent* names the

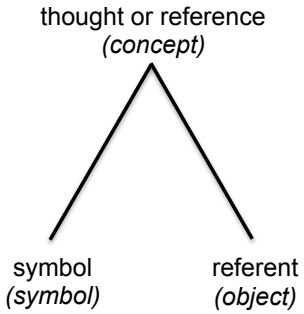


Figure 2.4: Semiotic triangle introduced by Ogden and Richards (1989)

object and the *thought or reference* is the concept. In contrast to de Saussure, the *concept* does not describe the conjunction of the *symbol* and *object*. Both the *symbol* and *object* are linked to the *concept*. Additionally, there is not a direct connection between the *symbol* and the *object*. According to Ogden and Richards (1989) the connection between the *object* and *symbol* is realized via the *concept*.

The concepts on how humans retrieve the meaning of language elements are mostly described for words. The philosopher Wittgenstein (1953) described the meaning of a word by the following definition:

> *For a large class of cases—though not for all—in which we employ the word "meaning" it can be defined thus: the meaning of a word is its use in the language. And the meaning of a name is sometimes explained by pointing to its bearer. (Wittgenstein, 1953, §43 translated to English by Gertrude Elizabeth Margaret Anscombe)*

Wittgenstein defines that the usage of a word determines its meaning. The usage again is specified by the context a word occurs and is used. Whereas Wittgenstein does not provide a specification of the context, the context can be e.g. words, sentences or paragraphs, which surround the word. In semantics, the study of meaning of a *symbol*, the contexts (e.g. sentences) are relevant and mostly topically coherent. Chapter 4 describes how text can automatically be split into topically coherent segments. Transferring the semiotic terminology to Wittgenstein's definition of word meaning, it can be translated into 'the meaning of a symbol is dependent on the symbols it is used with'. This requires relations between *symbols*.

De Saussure differentiates between two kinds of relations: *syntagmatic* and *associative* relations. *Syntagmatic* relations model linear relations between *symbols*, which can be e.g. a relation between words within a sentence or a syntactic relation. We assume these relations to be within topically coherent segments. This relationship is also called *in praesentia* as these symbols are present in the text. In Figure 2.5, the words *linguistics* and *manifestations* have a *syntagmatic* relation.

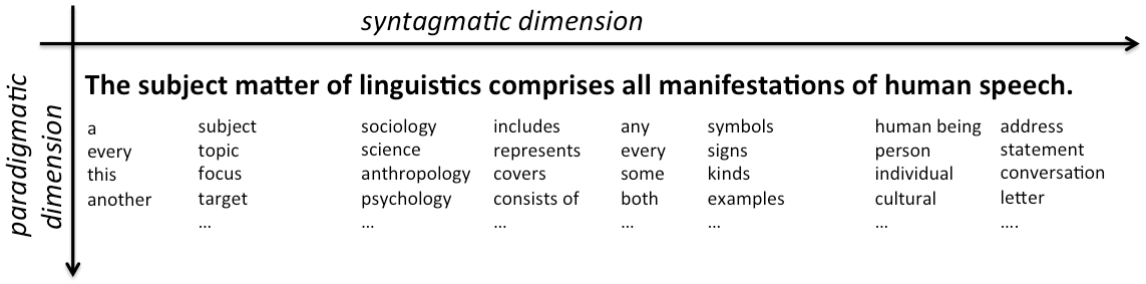


Figure 2.5: Exemplification of the metaphor of syntagmatic and paradigmatic relations.

The *associative* relation, also called *paradigmatic* relation, holds between words, which have something in common. Additionally, they do not have to be present in the text and are called *in absentia*. As shown in Figure 2.5, the term *linguistics* is associated with the terms *sociology* or *science*. Using these *paradigmatic* relations, we can provide suggestions for associated terms in the given context. To determine these relations de Saussure (1959) states:

> *Mental association creates other groups besides those based on the comparing of terms that have something in common; through its grasp of the nature of the relations that bind the terms together, the mind creates as many associative series as there are diverse relations. (de Saussure, 1959, p. 125)*

De Saussure claims that terms need to have something in common to form a *paradigmatic* relation. According to de Saussure this can be e.g. the word stem, a phonological similarity or associated terms. Based on these relations, we observe that words form a syntagmatic relation if they co-occur together within text and a paradigmatic relation if they share syntagmatic relations.

This led to the distributional hypothesis by Harris (1951), which was popularized by the definition of Firth (1957, p. 11): *"a word is characterized by the company it keeps"*. But following Firth (1957) the characterization is only represented by *syntagmatic relations*. Miller and Charles (1991, p. 4) define the term *contextual representation* as the "*knowledge of how [a] word is used*". In addition to *syntagmatic relations*, they also consider knowledge, which is not explicitly visible within the text. This knowledge is provided implicitly within the structure of text such as syntactic, semantic and pragmatic information.

Using a contextual representation of a language element (symbol), humans are able to understand its concept. This is supported by the ability of humans to know, which contexts within a larger unit e.g. sentences or documents are relevant for describing a language element. In accordance with Miller and Charles (1991, p. 5), we assume that this *contextual representation* is an "*abstract cognitive structure that accumulates from encounters with the word in various contexts*" as the cognitive context used by humans might be more general. We show the conversion of text into different *contextual representation* for language elements in Section 2.4.

Given this contextual knowledge Miller and Charles (1991, p. 8) define the *strong contextual* hypothesis: "Two words are semantically similar to the extent that their contextual representations are similar." According to this hypothesis we obtain higher semantic similarities the more context information two language elements share. This semantic similarity is also referred to as *distributional similarity*, as it is also based on the distributional hypothesis. We separate similarities into *topical similarities* (e.g. cat and claw), also called *relatedness*, and *near-synonymy* similarities (e.g. cat and tomcat). As presented in Figure 2.6, we expect to have a fluent transition between topical similarity and near-synonymy similarity. We expect to ob-

Figure 2.6: We expect a fluent transition for paradigmatic relations from topical similarity (relatedness) to near-synonymy similarity, which is dependent on the contextual representation used. Using wider context representations leads to more topical similarity, whereas using e.g. words within the same sentence yields paradigmatic relations, closer near-synonymy.

tain topical similarities when using context information, which is more related to the corpus or document and we expect near-synonymy similarities when using word and syntactic information, which is taken on word or sentence level. In Chapter 3 we survey different models for computing topical and near-synonymy similarities and introduce a method in Chapter 5, which uses general *contextual representations* to compute semantic similarities between language elements following a symbolic approach.

Whereas a symbol is used to describe a concept, there is no one-to-one mapping between symbols and concepts. For example the symbol *crane* has mappings to different concepts: It can be either the bird or the machine, which is used to move heavy objects. But according to the distributional hypothesis we collect *contextual representations* for both meanings of the term *crane*. Thus, using this representation to compute semantic similarities, we might obtain high similarities to the terms *digger* and *ostrich*. To separate different meanings we present a method in Section 8.3. Furthermore, the relation between *symbols* and *concepts* is not static. In language we observe that over time *symbols* can refer to different *concepts*, describe new *concepts* or do no longer refer to a *concept*. This can be exemplified for the term *sick*. Whereas it was used as a synonym for *ill*, today it can also be used as synonym for *amazing*. While we do not elaborate further on this phenomenon in this thesis, we have used our framework for the detection of these changes in (Mitra et al., 2014, 2015).

2.4 Context Extraction

Based on the contextual hypothesis (see Section 2.3), we can compute distributional similarities. For this, we first need to define the elements we want to use in order to compute similarities and the contextual representation. To ensure flexibility, we use *language elements*, as described in Section 2.1.1, as the elements, we compute similarities between. We use the term *context feature* to refer to the contextual representation[5] used for a *language element*.

In order to describe the contextual relation between a *language element* and a *context feature*, we use a tuple-based representation, which we define as follows:

Definition 1 (Contextual Relations). Let T be the set of all tuples $\langle x, y \rangle$, which describe the contextual relation between a *language element* $x \in X$ and a *context feature* $y \in Y$. X represents the set of all *language elements* and Y the set of all *context features*.

Our tuple-based notation is similar to the relation-based notation introduced by Curran (2004), but is more general as we do not restrict the *language elements* and *context features* to specific elements. This allows the usage of any arbitrary representation for *language elements* and *context features*. The notation of Curran (2004) is restricted to use terms as *language elements* and syntactic dependencies as *context features*.

To convert text into this tuple based presentation we use the *holing operation*. The holing operation is composed of two transformation functions: the *observation extraction* and the *holing function*. First, the *observation extraction* performs the linguistic preprocessing of raw text and delivers as output linguistic observations e.g. trigrams, dependency parses or POS tags. Then, the *holing function* is applied to each observation and returns a set of tuples. Thus, it is independent of the tools and methods used for the observation generation. This allows

[5] We consider contextual representation as syntagmatic relations including implicit knowledge e.g. dependency parses as described in Section 2.3

Sentence:	I	gave	a	book	to	the	girl
Positions:	1	2	3	4	5	6	7

Figure 2.7: This figure shows an example sentence, which is used to explain the workings of different context extraction methods.

using the same *holing operation* used for extracting e.g. syntactic dependencies computed with various implementations.

We show examples of different holing operations based on the sentence shown in Figure 2.7. First, we introduce the observation extraction and give examples for different preprocessing steps.

2.4.1 Observation Extraction

The observation extraction function f_{obs} is performed on raw text and results to n observations $o_1, \ldots o_n \in O$. Each observation o_i is a list of structures extracted from the text. The transformation function can thus be formalized as follows:

$$f_{obs}(raw\ text) \Rightarrow \{o_1, \ldots, o_n\} \tag{2.5}$$

As first example we perform an observation extraction, based on the example sentence, considering collapsed syntactic dependencies computed with the Stanford Parser (de Marneffe et al., 2006). This results to the collapsed syntactic dependency parse as shown in Figure 2.8.[6]

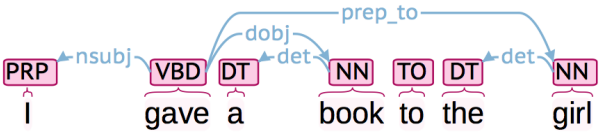


Figure 2.8: This figure shows the collapsed dependency parse computed with Stanford Parser for the example sentence. Above each term the according POS is written. On top of the POS we observe the dependency relations detected by the parser.

Each observation o_i is a list containing a dependency relation r and the two terms t_i, t_j connected by this relation, which leads to observations in the format $(r; t_i; t_j)$. We formalize the observation transformation function in Algorithm 1:

As syntactic dependency parses provide a lot of structural information, we expect near-synonymy similarities when used for computing similarities. Applying Algorithm 1 to the parsed example sentence shown in Figure 2.8 results to the following observations:

[6]Due to the collapsed dependencies there is no relation for the term *to* in Figure 2.8. However, the context representation for content words results to a more meaningful representation when using collapsed dependencies and not all prepositions are collapsed.

Algorithm 1 Algorithm for extracting observations from syntactic dependencies

```
function OBSERVATION(sentences)                    ▷ sentences: list of parsed sentences
    observations = {}
    for s ∈ sentences do                                          ▷ Iterate each sentence s
        for t_i ∈ s do                                  ▷ Iterate each token t_i in sentence s
            for t_j ∈ sentence do                       ▷ Iterate each token t_j in sentence s
                if relation r between t_i and t_j then   ▷ Check if t_i and t_j have a relation
                    observations.add((r;t_i;t_j))                              ▷ add observation
    return observations
```

a) collapsed dependency parse:

$(\texttt{nsubj;gave}_2\texttt{;I}_1\texttt{)}$, $(\texttt{det;book}_4\texttt{;a}_3\texttt{)}$, $(\texttt{dobj;gave}_2\texttt{;book}_4\texttt{)}$,
$(\texttt{det;girl}_7\texttt{;the}_6\texttt{)}$, $(\texttt{prep\_to;gave}_2\texttt{;girl}_7\texttt{)}$,

As most dependency parsers require or predict POS tags, we can incorporate this information to the term without any additional computational efforts. Using POS information results to similarities, which are more targeted to terms of the same POS. This is relevant for terms that occur with different POS, e.g. the term *drive* can be a verb or a noun. In order to avoid similarities of the same word with different morphological variations (e.g. *pitcher* might be most similar to *pitchers*), terms can be additionally lemmatized. This step also reduces the size of the vocabulary that needs to be processed for the similarity computation. For the lemmatized dependency observation extraction we lemmatize terms and append POS tags. Due to the lemmatization, we also need to truncate the POS tags for nouns, adjectives and verbs. To keep proper nouns we change the POSs tags *NNPS* and *NNP* to *NP* and truncate the POS tags for all further nouns, adjectives and verbs to the length of two characters. All remaining POS tags keep the same. This results to the observation format (*relation*, $lemma\#pos_1$, $lemma\#pos_2$) and is exemplified next:

b) collapsed dependency parse with lemmas and POS tags:

$(\texttt{nsubj;give\#VB}_2\texttt{;I\#PRP}_1\texttt{)}$, $(\texttt{det;book}_4\texttt{;a\#DT}_3\texttt{)}$, $(\texttt{dobj;give\#VB}_2\texttt{;book\#NN}_4\texttt{)}$,
$(\texttt{det;girl\#NN}_7\texttt{;the\#DT}_6\texttt{)}$, $(\texttt{prep\_to;give\#VB}_2\texttt{;girl\#NN}_7\texttt{)}$,

A simpler preprocessing step splits the sentence into token n-grams, which does not require any language-dependent preprocessing. Considering the n-gram extraction each observation is a list of N tokens, as described in Algorithm 2.

First, we show the observation output when using a bigram ($N = 2$) in Enumeration *c*:

c) bigram:

$(\texttt{I}_1\texttt{;gave}_2\texttt{)}$,$(\texttt{gave}_2\texttt{;a}_3\texttt{)}$, $(\texttt{a}_3\texttt{;book}_4\texttt{)}$, $(\texttt{book}_4\texttt{;to}_5\texttt{)}$, $(\texttt{to}_5\texttt{;the}_6\texttt{)}$, $(\texttt{the}_6\texttt{;girl}_7\texttt{)}$.

Algorithm 2 Algorithm for extracting n-gram observations

```
function OBSERVATION(text,N)          ▷ text: list of tokens, N: number of tokens
    observations= {}
    for i = 0; i < |text| − N; i + + do
        observation=String[N]          ▷ Create an array of N elements
        for j = 0; j < N; j + + do
            observation[j]=text[i+j]   ▷ add token to the array
        observations.add(observation)
    return observations
```

We present the output for the trigram observations in the next example using the same algorithm as for the bigram extraction. In contrast to the bigram observation, three tokens ($N = 3$) are extracted for each observation, as shown in the subsequent observation output:

d) trigram:
($\$_0$;$I_1$;$gave_2$), ($I_1$;$gave_2$;$a_3$), ($gave_2$;$a_3$;$book_4$), ($a_3$;$book_4$;$to_5$), ($book_4$;$to_5$;$the_6$), ($to_5$;$the_6$;$girl_7$), ($the_6$;$girl_7$;$\$_8$) .

Furthermore, we are also interested in terms, which consist of more than a single word. To achieve this, we extend the trigram observation to consider terms as *language element* with more than one token. We realize this by defining the term in the middle of the trigram to be an n-gram itself with length $K = 1 ... K$, as described in Algorithm 3.

Algorithm 3 Algorithm for extracting trigram observations with n-grams as second element

```
function (text, K)          ▷ text: list of tokens, K: length of the n-gram in the middle
    observations={}
    for i = 0; i < |text|; i + + do
        observation=String[3]          ▷ observation has three elements
        observation[0]=(i>0)?text[i-1]:‘ ’   ▷ set left token
        ngram = text[i]                ▷ generate n-gram of length K
        for j = 1; j < K; j + + do
            ngram+=‘ ’+text[i+j]
        observation[1]=ngram           ▷ set the n-gram
        observation[2]=(i>=|text|)? ‘ ’:text[i+1]   ▷ set the right token
        observations.add(observation)
    return observations
```

For ease of the observation example, we restrict the n-gram in the middle to be up to length two ($K = 2$). This results to the observations shown below:

e) trigram with n-grams at the second position:
($\$_0$;$I_1$;$gave_2$),($\$_0$;$I_1$ $gave_2$;the_3), (I_1;$gave_2$;a_3), (I_1;$gave_2$ a_3;$book_4$),

(gave$_2$;a$_3$;book$_4$),(gave$_2$;a$_3$ book$_4$; to$_5$), (a$_3$;book$_4$;to$_5$), (a$_3$;book$_4$ to$_5$;the$_6$),
(book$_4$;to$_5$;the$_6$), (book$_4$;to$_5$ the$_6$;girl$_7$), (to$_5$;the$_6$;girl$_7$), (to$_5$;the$_6$ girl$_7$;$_8$)
(the$_6$;girl$_7$;\$$_8$) .

Whereas the observation extraction can be extended into further directions, we have concentrated on observations, which are also used in this thesis. In the next section, we explain the holing transformation step. This step converts observations into the tuple representation that is used to compute similarities between *language elements*.

2.4.2 Holing Operation

Here we define the *holing operation*, which performs the transformation of text into tuples $\{\langle x_1, y_1\rangle ..., \langle x_n, y_m\rangle\}$. For this, it uses the output of the observation extraction and applies a *holing function*, which operates on a single observation. We formalize the holing function f_{holing}, which transforms a single observation into a set of tuples:

$$f_{holing}(o) \Rightarrow \{\langle x_1, y_1\rangle ..., \langle x_n, y_m\rangle\} \tag{2.6}$$

The *holing operation* iterates over all observations and applies the holing function in order to obtain the tuple-based output.

To split an observation into a tuple we place the *hole symbol* "@" at the position of the element that is used as *language element* and use this modified observation as *context feature*. Using the dependency parse observations *a)* and *b)* from the previous section an observation is transformed using the following function:

$$f_{holing}((r; t_1; t_2)) = \{\langle t_1, (r; @; t_2)\rangle, \langle t_2, (r; t_1; @)\rangle\} \tag{2.7}$$

As dependency relations are drawn between two terms, we can use each terms as *language element* and the relation including the remaining term as *context feature*. We show the output of the so-called *dependency parse holing operation* output for the observations *a)* and *b)* in the following enumerations:

a) collapsed dependency parse:
⟨I$_1$,(nsubj;gave$_2$;@)⟩, ⟨gave$_2$,(nsubj;@;I$_1$)⟩, ⟨a$_3$,(det;book$_4$;@)⟩,
⟨book$_4$,(det;@;a$_3$)⟩, ⟨gave$_2$,(dobj;@;book$_4$)⟩, ⟨book$_4$,(dobj;gave$_2$;@)⟩,
⟨girl$_7$,(det;@;the$_6$)⟩, ⟨the$_6$,(det;girl$_7$;@)⟩, ⟨gave$_2$,(prep_to;@;girl$_7$)⟩,
⟨girl$_7$,(prep_to;gave$_2$;@)⟩, .

b) collapsed dependency parse with lemmas and POS tags:
⟨I#PRP$_1$,(nsubj;give#VB$_2$;@)⟩, ⟨give#VB$_2$,(nsubj;@;I#PRP$_1$)⟩,
⟨a#DET$_3$,(det;book#NN$_4$;@)⟩, ⟨book#NN$_4$,(det;@;a#DET$_3$)⟩,

$\langle$(dobj;give#VB$_2$;book#NN$_4$)$\rangle$, $\langle$(dobj;give#VB$_2$;book#NN$_4$)$\rangle$,
$\langle$girl#NN$_7$,(det;@;the#DT$_6$)$\rangle$, $\langle$the#DT$_6$,(det;girl#NN$_7$;@)$\rangle$,
$\langle$give#VB$_2$,(prep_to;@;girl#NN$_7$)$\rangle$, $\langle$girl#NN$_7$,(prep_to;give#VB$_2$;@)$\rangle$,

For the holing output of *b)* we show a graphical presentation in the left graph in Figure 2.9.[7] Whereas the graph is similar to the one shown above in Figure 2.8, we have two relations be-

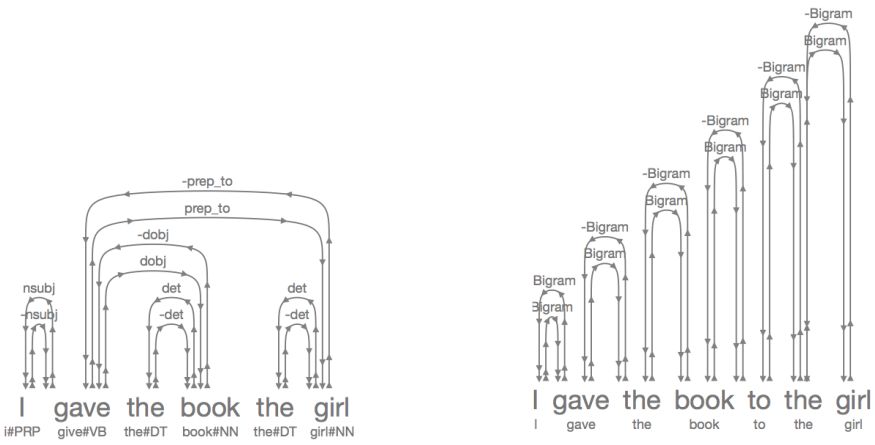


Figure 2.9: On the left graph we show the visualization of the *Stanford collapsed dependency parsing holing operation* using lemmas and POS. We add a minus to the relation if the hole is placed into the first position and thus the relation is "inverse". The graph on the right shows the *bigram holing operation* based on the example sentence. As relation name we display the holing operation name (*Bigram*) and add a minus sign to mark negative relations.

tween the terms, as we consider both terms in the observation as *language element*. Contrary to the holing output in the enumeration, we mark the direction of the relation appending a minus "-" to the relation name. In the remainder of this thesis we will refer to the *lemmatized collapsed dependency parse holing operation* as *collapsed dependency parse holing operation*.

Using the bigram observations from *c)* the hole can similar to the dependency parse holing function be placed at two positions. This results to two tuples using the following transformation function:

$$f_{holing}((t_1, t_2)) = \{\langle t_1, (@; t_2)\rangle, \langle t_2, (t_1; @)\rangle\} \tag{2.8}$$

When we apply this function to the bigram observations, we achieve the following tuples:

c) bigram:
$\langle$I$_1$,(@;gave$_2$)$\rangle$, $\langle$gave$_2$,(I$_1$;@)$\rangle$, $\langle$gave$_2$,(@;a$_4$)$\rangle$, $\langle$a$_3$,(@;book$_4$)$\rangle$, $\langle$book$_4$,(a$_3$;@)$\rangle$,
$\langle$book$_4$,(@;to$_5$)$\rangle$, $\langle$to$_5$,(book$_4$;@)$\rangle$, $\langle$to$_5$,(@;the$_6$)$\rangle$, $\langle$the$_6$,(to$_5$;@)$\rangle$, $\langle$the$_6$,(@;girl$_7$)$\rangle$,
$\langle$girl$_7$,(the$_6$;@)$\rangle$.

[7] The graphs are generated using the JoBimViz demo (Ruppert et al., 2015a), which is accessible online: `http://maggie.lt.informatik.tu-darmstadt.de/jobimviz/`

A graphic representation of the *bigram holing operation* is illustrated in the right graph in Figure 2.9. Whereas this *bigram holing operation* considers the position of the *language elements*, we could also simplify the holing function to handle left and right neighbors equally:

$$f_{holing}((t_1, t_2)) = \{\langle t_1, t_2\rangle, \langle t_2, t_1\rangle\} \tag{2.9}$$

Using the *trigram holing operation*, we decide to place the hole only into the second position. This aims to achieve a precise context representation, as we use both the left and the right term as one *context feature*. The function transforms one observation into a single tuple and then applies the following holing function:

$$f_{holing}((t_1, t_2, t_3)) = \{\langle t_2, (t_1; @; t_3)\rangle\} \tag{2.10}$$

Applying the *trigram holing operation* results to the following tuples:

d) trigram:
$\langle \mathtt{I}_1, (\$_0; @; \mathtt{gave}_2)\rangle$, $\langle \mathtt{gave}_2, (\mathtt{I}_1; @; \mathtt{a}_3)\rangle$, $\langle \mathtt{a}_3, (\mathtt{gave}_2; @; \mathtt{book}_4)\rangle$, $\langle \mathtt{book}_4, (\mathtt{a}_3; @; \mathtt{to}_5)\rangle$,
$\langle \mathtt{to}_5, (\mathtt{book}_4; @; \mathtt{the}_6)\rangle$, $\langle \mathtt{the}_6, (\mathtt{to}_5; @; \mathtt{girl}_7)\rangle$, $\langle \mathtt{girl}_7, (\mathtt{the}_6; @; \$_8)\rangle$.

The graphical representation of the *trigram holing operation* is shown in Figure 2.10.

The *n-gram-based trigram holing operation*, which supports the usage of n-grams as *language elements*, is processed similar to the single-worded trigram representation but the term at the second position is an n-gram. We present the output of this holing operation, using a bigram ($K = 2$), in the right graph in Figure 2.10. Additionally, the textual representation of the 13 tuples is illustrated in the following enumeration:

e) trigram with n-gram at the second position:
$\langle \mathtt{I}_1, (\$_0; @; \mathtt{gave}_2)\rangle$, $\langle \mathtt{I}_1\ \mathtt{gave}_2, (\$_0; @; \mathtt{the}_3)\rangle$, $\langle \mathtt{gave}_2, (\mathtt{I}_1; @; \mathtt{a}_3)\rangle$,
$\langle \mathtt{gave}_2\mathtt{a}_3, (\mathtt{I}_1; @; \mathtt{book}_4)\rangle$, $\langle \mathtt{a}_3, (\mathtt{gave}_2; @; \mathtt{book}_4)\rangle$, $\langle \mathtt{a}_3\ \mathtt{book}_4, (\mathtt{gave}_2; @; \mathtt{to}_5)\rangle$,
$\langle \mathtt{book}_4, (\mathtt{a}_3; @; \mathtt{to}_5)\rangle$, $\langle \mathtt{book}_4\ \mathtt{to}_5, (\mathtt{a}_3; @; \mathtt{the}_6)\rangle$, $\langle \mathtt{to}_5, (\mathtt{book}_4; @; \mathtt{the}_6)\rangle$,
$\langle \mathtt{to}_5\ \mathtt{the}_6, (\mathtt{book}_4; @; \mathtt{girl}_7)\rangle$, $\langle \mathtt{the}_6(\mathtt{to}_5; @; \mathtt{girl}_7)\rangle$, $\langle \mathtt{the}_6\ \mathtt{girl}_7, (\mathtt{to}_5; @; \$_8)\rangle$
$\langle \mathtt{girl}_7, (\mathtt{the}_6; @; \$_8)\rangle$

The indices appended to the *language elements* and *context features* are used to count *language elements* and *context features* correctly and avoid counting them more than once. This might happen for the term *book* using the *syntactic dependency holing operation* as it occurs three times as *language element* in the tuple representation but only once within the text.

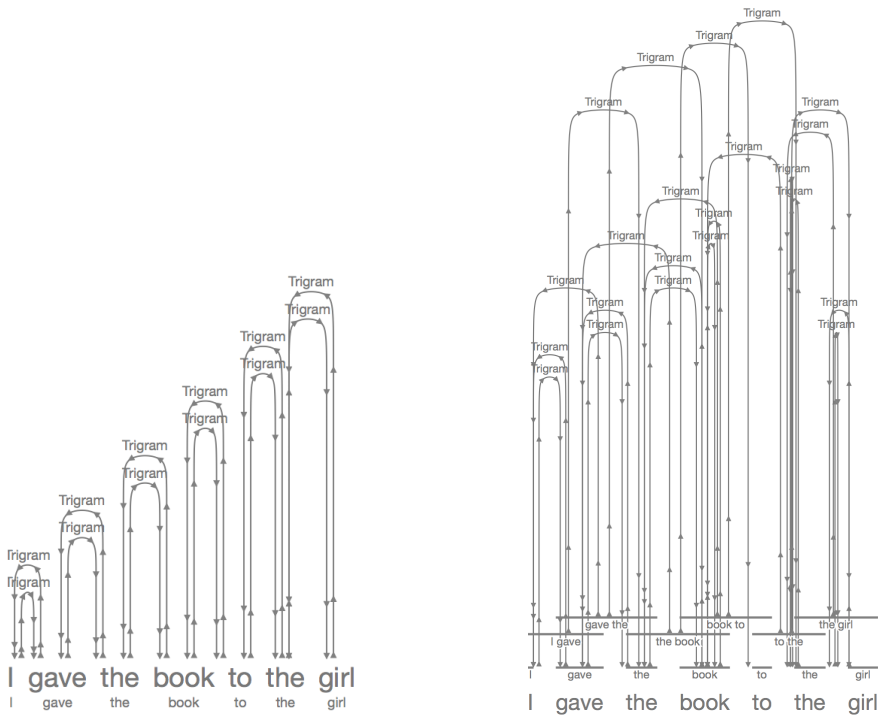


Figure 2.10: Graphical representation of the trigram holing operation based on the example sentence considering only single words (left) and n-grams up to length two (right). Each word/n-gram has two outgoing links, which are labeled with Trigram. These links refer to the left and right neighboring terms, which represent the context feature.

2.5 Graph-based Notation

Using the tuple-based notation for *language elements* and *context feature* (see Definition 1 in Section 2.4) enables a flexible symbolic notation, which allows a graph-based notation as defined in the following:

Definition 2. Let G be a graph of the finite set (N, E) with nodes N and edges E. The nodes $N = X \cup Y$ are composed of *language elements* X and *context features* Y. The edges E are defined by the tuples $\langle X, Y \rangle \in E$.

Whereas we describe *language elements* as the set X and *context features* as Y, they can be switched in the tuple representation. This allows computing similarities between *context features*. Next, we define the cardinality of X and Y describing the number of *language elements* and *context features*.

Definition 3 (Cardinality). The cardinality of the set of *language elements* X and the set of *context features* Y is defined by $|.|$. The number of unique *language elements* is defined by $|X|$ and the number of *context feature* is expressed using $|Y|$. The number of all tuples $\langle x, y \rangle \in T$ is defined by $|T|$.

The next definition defines the notation for determining frequencies of x and y and their co-occurrences.

Definition 4 (Frequencies). In contrast to the cardinality, for defining the frequency we use the tuple notation inside the squared bracket notation. The frequency for the co-occurrence of $x \in X$ and $y \in Y$ is defined by $|\langle x, y\rangle|$. Using the asterisk $*$ as placeholder for every occurrence of either x and y, $|\langle x, *\rangle|$ return the frequency of the *language element* x and $|\langle *, y\rangle|$ the frequency of the *context feature* y. Using $|\langle X, *\rangle|$ returns the frequency of all *language elements*. Similarly the frequency of all *context features* Y is retrieved by $|\langle *, Y\rangle|$. The frequency for all different co-occurrences of *language elements* with *context feature* is retrieved with $|\langle X, Y\rangle|$ and $|\langle *, *\rangle|$.

In order to rank tuples, we introduce a partial order in the next definition, which is based on a scoring function:

Definition 5 (Partial Ordering). Let $s(.)$ be a function for scoring tuples. Scoring tuples $\langle x, y\rangle \in T$ according to $s(.)$ results to a partially ordered set with the property $s(\langle x_i, y_j\rangle) \leq s(\langle x_k, y_l\rangle)$ with $\langle x_i, y_j\rangle, \langle x_k, y_l\rangle \in T$.

Using a partial order between tuples allows retrieving the highest ranked tuples, which we define next:

Definition 6 (Retrieve the highest scored tuples). Let S and R be a subset of all tuples $\langle x, y\rangle \in T$ with the property $S = T \cap R$ and $s(.)$ be a scoring function for tuples T, which results to a partial ordering. The set S contains the highest n scored tuples if $\forall \langle x_i, y_j\rangle, \langle x_k, y_l\rangle : \langle x_i, y_j\rangle \in S \wedge \langle x_i, y_j\rangle \in R \wedge s(\langle x_i, y_j\rangle) \geq s(\langle x_k, y_l\rangle) \wedge |S| = n$. The highest n scored tuples for a *language element* x can be retrieved with $highest(x, n) = S$.

The introduced definitions enable the basis for a flexible and symbolic representation for computing similarities. We use the notations in the following chapters to describe the methods.

2.6 Zipf's Law

Observing the distribution of speech units, Zipf (1929) discovered the phenomenon that language follows a power-law distribution. This principle is called Zipf's law and does not only apply for speech but also for textual data. Considering text it states that the term frequency f is proportional to its rank r when ranked in a decreased order according to the frequency: $f \propto \frac{1}{r}$.

According to this law, few terms in text occur very frequently, which are mainly stopwords. Additionally, there are many terms within the text, which have very low frequencies. In Figure 2.11, we show the term frequency against the rank in log-scale (see triangles).

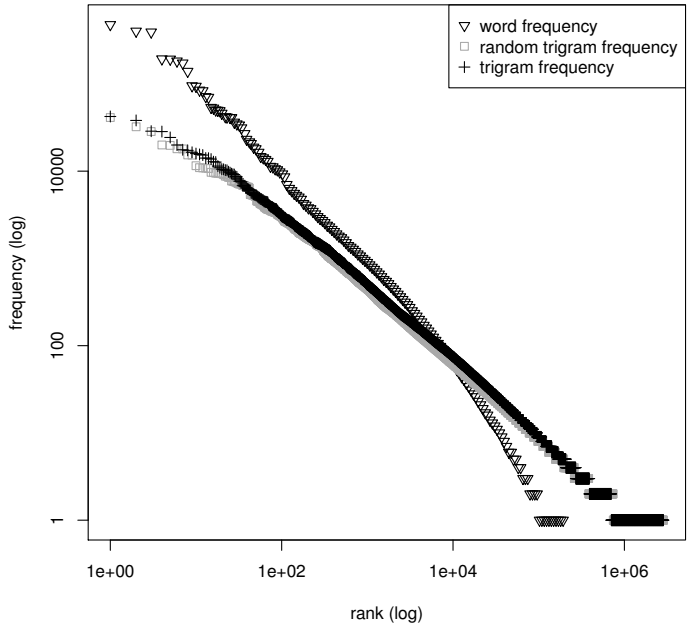


Figure 2.11: In this graph we plot the frequency against the inverse rank according to the frequency, based on a log-log scale. We draw the word frequency, the trigram context features and the trigram random context features, which are two words randomly extracted from a sentence.

However, the law does not only apply to terms but also to contextual representations. It is obvious that this holds for the bigram holing operation (see Section 2.4.2), as it is equivalent to the term-based frequencies.

Also, using the left and right neighboring terms as *context features*, called trigram holing (see Section 2.4.2), follows a power-law distribution, as shown with the crosses in Figure 2.11. Whereas the frequency is lower, we also observe more elements. According to the distribution of the *context features*, we assume that whereas high frequent context features are too general to be used for similarity computation, infrequent ones contribute only marginally for the usage of similarity computations. Thus, it seems to be reasonable to use mid-frequent context features, which is demonstrated in Chapter 5. We also plot the distributional of a "random trigram" context extraction (see gray squares). The random context is generated selecting two random words from the sentence as *context feature*. We observe almost the same distribution as using the regular trigram context. This shows that even randomly selected *context features* follow Zipf's law. Biemann (2011) also verified the adaptability of the power-law to various *language elements* for e.g. letter n-grams, word n-grams, sentences and search queries. Thus, we expect any arbitrary contextual representation used for any *language element* to follow Zipf's Law.

2.7 Weighting Language Elements and Context Features

Weighting function applied to frequencies of *language elements* and *context features* are used in NLP, to give elements providing less information (e.g. stopwords or function words) lower scores and to give higher scores to content relevant elements. One of the most obvious weighting is the frequency of a *language element* or a *context feature*. However, using the frequency ranks mostly stopwords highest (see Zipf's Law in Section 2.6). First, we start with mutual information measures, which give higher scores to pairs of *language elements* and *context features* that do not co-occur randomly. Thus, we also call them significance measurements. Afterwards, we introduce the tf-idf, which is restricted to be computed to a term-document representation. Further measures not introduced in this thesis can be found in (Manning et al., 2008, p. 100).

2.7.1 Mutual Information Measures

According to our definition in Section 2.5 we use x to represent a *language element* and y for a *context feature*. The mutual information measures rely on the frequencies of *language elements* $|\langle x, * \rangle|$, *context features* $|\langle *, y \rangle|$ and the combination of both of them are $|\langle x, y \rangle|$. Here, we introduce three mutual information measures Pointwise Mutual Information (PMI), Lexicographer's Mutual Information (LMI) and Log-Likelihood (LL):

- Pointwise Mutual Information (PMI): a widely used significance measure since its introduction to NLP by Church and Hanks (1990).
- Lexicographer's Mutual Information (LMI) (Bordag, 2008), also referred to as Local Mutual Information: since PMI is known to assign high significance scores to pairs formed by low-frequent items, the LMI measure tries to balance this by multiplying the PMI score with the pair frequency.
- Log-Likelihood (LL): the LL ratio test is a widely used measure since its introduction by Dunning (1993) and compares the ratio between two parametrized distributions. It is less susceptible in the overestimation of low frequency pairs. The transformation, shown in Table 2.1, was introduced by Bordag (2008) and avoids problems observed with extreme probability values.

Additionally, we show the formulas of all three measures in Table 2.1.

2.7.2 tf-idf

One of the standard weighting method is the so-called *tf-idf* (see e.g. (p. 119 Manning et al., 2008)). In contrast to the previously measures, it is restricted to a term-document representation. It is motivated by the observation that words occurring in many documents within a

Name	Equation	
PMI	$PMI(\langle x, y\rangle) =$	$\log_2(\lvert\langle *, *\rangle\rvert \cdot \frac{\lvert\langle x,y\rangle\rvert}{\lvert\langle x,*\rangle\rvert \cdot \lvert\langle *,y\rangle\rvert})$
LMI	$LMI(\langle x, y\rangle) =$	$\lvert\langle x, y\rangle\rvert \cdot \log_2(\lvert\langle *, *\rangle\rvert \cdot \frac{\lvert\langle x,y\rangle\rvert}{\lvert\langle x,*\rangle\rvert \cdot \lvert\langle *,y\rangle\rvert})$
LL	$\lambda =$	$\lvert\langle *, *\rangle\rvert \log \lvert\langle *, *\rangle\rvert - \lvert\langle x, *\rangle\rvert \log \lvert\langle x, *\rangle\rvert - \lvert\langle *, y\rangle\rvert \log \lvert\langle *, y\rangle\rvert$ $+\lvert\langle x, y\rangle\rvert \log \lvert\langle x, y\rangle\rvert$ $+(\lvert\langle *, *\rangle\rvert - \lvert\langle x, *\rangle\rvert - \lvert\langle *, y\rangle\rvert + \lvert\langle x, y\rangle\rvert)$ $\cdot \log(\lvert\langle *, *\rangle\rvert - \lvert\langle x, *\rangle\rvert - \lvert\langle *, y\rangle\rvert + \lvert\langle x, y\rangle\rvert)$ $+(\lvert\langle x, *\rangle\rvert - \lvert\langle x, y\rangle\rvert) \cdot \log(\lvert\langle x, *\rangle\rvert - \lvert\langle x, y\rangle\rvert)$ $+(\lvert\langle *, y\rangle\rvert - \lvert\langle x, y\rangle\rvert) \cdot \log(\lvert\langle *, y\rangle\rvert - \lvert\langle x, y\rangle\rvert)$ $-(\lvert\langle *, *\rangle\rvert - \lvert\langle x, *\rangle\rvert) \cdot \log(\lvert\langle *, *\rangle\rvert - \lvert\langle x, *\rangle\rvert)$ $-(\lvert\langle *, *\rangle\rvert - \lvert\langle *, y\rangle\rvert) \cdot \log(\lvert\langle *, *\rangle\rvert - \lvert\langle *, y\rangle\rvert)$
	$LL(\langle x, y\rangle) =$	$\begin{cases} -2 \log \lambda & \text{if } \lvert\langle x, y\rangle\rvert < \frac{\lvert\langle x,*\rangle\rvert \lvert\langle *,y\rangle\rvert}{\lvert\langle *,*\rangle\rvert} \\ 2 \log \lambda & \textit{otherwise} \end{cases}$

Table 2.1: Mutual information measures used to weight *language elements* and *context features* using the graph notation from Section 2.5.

corpus are less important. On the contrary, terms that are frequent and occur only in some documents tend to be more important terms.

The weighting measure is a combination of the term frequency and the so-called inverse document frequency. Following our definition we assume a term x as *language element* and a document y as *context feature*. The document frequency $df(x)$ measures in how many documents a term occurs and is described by the following formula:

$$df(t) = \sum_{y_i : \lvert\langle x, y_i\rangle\rvert > 0}^{Y} 1 \tag{2.11}$$

The inverse document frequency is the logarithmic inverse relative document frequency *idf*. Using the logarithm gives a weight of zero to terms, which occur in all documents and a low score to terms occurring in many documents. The *tf-idf* is the multiplication of the term frequency and the inverse document frequency leading to:

$$\text{tf-idf}(x, y) = \lvert\langle x, y\rangle\rvert \cdot \log \frac{|Y|}{df(x)} \tag{2.12}$$

We give a small example for the tf-idf computation based on a collection of three documents in Table 2.2 We observe that the term "*the*" occurs in three documents. Thus, it has an inverse document frequency of zero and has a *tf-idf* value of 0.0. As the term *hamster* occurs in two out of three documents, the *idf* is below one. For the term *plane* the *tf-idf* increases in comparison to the initial term frequency as the *idf* is above one.

word	doc. 1	doc. 2	doc. 3		doc. 1	doc. 2	doc. 3
	tf			idf	tf-idf		
the	5	7	10	0.00	0.00	0.00	0.00
hamster	12	11	0	0.41	4.92	4.51	0.00
plane	0	0	10	1.10	0.00	0.00	11.00

Table 2.2: Example for *tf-idf* scores based on a document-term matrix.

2.8 Computing Similarities

In order to compute similarities between *language elements*, measures are required, which determine the similarities between the *context features* of two *language elements*. The most common similarity measure used in NLP is the *cosine similarity*, which measures the angle between *context feature* vectors. For the computation we consider a score for each tuple $\langle x, y \rangle$, denoted by $s(\langle x, y \rangle)$. The simplest scoring would be the usage of the frequency of the co-occurrence, but we could also use the mutual information measures as score. The cosine can then be computed using the following formula:

$$cosine(x_i, x_j) = \frac{\sum_{k=1}^{|Y|} s(\langle x_i, y_k \rangle) \cdot s(\langle x_j, y_k \rangle)}{\sqrt{\sum_{k=1}^{|Y|} s(\langle x_i, y_k \rangle)^2 \cdot \sum_{k=1}^{|Y|} s(\langle x_j, y_k \rangle)^2}}. \tag{2.13}$$

Computing the cosine between two *language elements* results to scores between -1 and 1. A score of one indicates high similarity between the context representations of both *language elements*. No similarity between two *language elements* is signified with a score of zero. Considering frequency counts, which are always positive, negative scores are not possible. The formula describes how to compute the similarities based on *language elements*, but the formula can easily be changed to compute similarities between *context features* (e.g. documents) by switching X and Y.

Another similarity metric is the Jaccard measure (Jaccard, 1912). In contrast to the cosine similarity, the Jaccard similarity does not consider a numeric vector representation but computes the overlap of *context features* for two *language elements*. It computes the intersection of context feature vectors divided by the union of the features of two *language elements*. For the computation we could consider the co-occurrence of *language elements* and *context features* (frequency above zero) or some scoring function $s(\langle x, y \rangle)$ as shown in the formula below:

$$jaccard(x_i, x_j) = \frac{\sum_{y}^{Y=Y_i \cap Y_j | s(\langle x_i, Y_i \rangle)>0 \wedge s(\langle x_i, Y_i \rangle)>0} 1}{\sum_{y}^{Y=Y_i \cup Y_j | s(\langle x_i, Y_i \rangle)>0 \vee s(\langle x_i, Y_i \rangle)>0} 1}. \tag{2.14}$$

We do not show the Euclidean metric, which computes similarity based on vector length. This measure is not appropriate for high dimensional and sparse vectors (Strehl et al., 2000), as

for similarity computations the position within the vectors is more important than the vector length. Further measures used for computing distributional similarities can be found in e.g. (Lee, 1999; Kiela and Clark, 2014).

Chapter 3

The Meaning of a Word in a Nutshell

> "There are at least as many meanings of 'meaning' as there are disciplines, which deal with language, and of course, many more than this because exponents within disciplines do not always agree with one another."
>
> (see Osgood et al., 1957, p. 2)

Statistical Semantics is a sub-field in NLP, which makes use of statistical methods to identify the meaning of *language elements* (i.e. tokens, terms, sentences, paraphrases or documents) based on the contextual representations (i.e. neighboring words, dependency parses, vector representations), described in the previous chapters.

The usage of mathematical approaches in linguistics started with counting the frequencies of words in different languages e.g. Kaeding (1898); Noreen (1906). The first so-called statistical methods applied to the field of linguistics where proposed by Zipf (1929) (see Section 2.6). His findings focused on the frequencies of phonemes but also hold true for written natural language. Nowadays these studies are categorized into the field of quantitative analysis. Nevertheless, they can be considered as seminal work for the introduction of statistical methods in the field of NLP. The term *statistical semantics* was first mentioned in a theoretic description of machine translation introduced by Weaver (1949/1955). He discusses the problem of different meanings of a word according to its context, which is summarized in (Malmberg, 1964, p. 202) as: "the machine must [...] be able to choose the correct alternative [term], which involves 'learning' that in one context one alternative [term] is required and in another context a different alternative [term]". For the selection of correct answers, not only dictionaries are deemed necessary but also the information, which terms are similar in the given context. To achieve this goal, manually created lexical resources can be used e.g. WordNet (Miller, 1995) or manually created thesauri (e.g. Roget's Thesaurus (Berry, 1962) or Merriam-Webster's Thesaurus

(Merriam-Webster, 1994)). However, manually created resources are mostly incomplete, as covering all words of a language is impossible. Additionally, language changes over time and word senses emerge and disappear over time as we have shown in (Mitra et al., 2014). A further issue is the changing meaning of terms depending on the domain they are used. In addition, the creation process and the maintenance of such resources is time intensive. To alleviate the efforts associated with the manually creation of lexical resources, methods are required, which determine the meaning of a term based on patterns within text.

Rubenstein and Goodenough (1965) introduced one of the first studies, confirming the distributional hypothesis. Considering the amount of co-occurring words two words share, they computed the relatedness. They validated their approach against 65 word pairs, which were manually rated according to their similarity. The mean between several ratings for a word pair were then considered as gold standard.

One of the first approaches for computing term similarity used as context representation neighboring character bigrams of a term (Adamson and Boreham, 1974). This can be attributed to the lack of computational resources. The need to compute similarities between terms, sentences or documents became relevant in the field of information retrieval, which evolved in the 1960s. In the first decades, the main challenge in this field was the retrieval of relevant documents to a given query. As user queries often contain terms that do not occur within the relevant articles, techniques are needed that allow determining how similar two terms are.

One of the first approaches was the representation of documents as vectors of words leading to the vector space model (VSM) (Salton et al., 1975; Dubin, 2004; Schütze, 1993). This representation allows computing similarities between documents by direct comparison or clustering. Next, similarities to the related documents can be computed by using the centroids of each cluster. The VSM is not only applied to compute similarities between documents but also between words. The first approach, which was used to represent a term as an abstract vector representation, was introduced by Rumelhart et al. (1988). Deerwester et al. (1990) introduced Latent Semantic Analysis (LSA), an algebraic approach, which allows transforming documents and terms from a "term"-feature space into a more abstract concept space. This space captures semantic information relying on the Eigenvectors of the VSM, which offers a numeric vector representation. Thus, similarities between documents and terms can be computed. Whereas the main goal of using LSA is the reduction of the context representation, it also initiated the research for the so-called Topic Models (TMs), which capture the semantic relatedness from a topical aspect. The two most prominent topic models followed with the probabilistic Latent Semantic Analysis (pLSA) (Hofmann, 1999, 2001) and Latent Dirichlet Allocation (LDA) (Blei et al., 2003).

In the remainder of this chapter, we give some insight into different models used to compute similarities between terms and documents. We classify these methods into the two categories:

– document-based semantic models

– term-based semantic models

Document-based models are computed on the basis of a term-document representation and mostly do not consider word positions or distance relations between words. Thus, they mainly perform on co-occurrences of words within a document. These models are mainly used to compute similarities between documents, but are also capable of computing similarities between terms. As they operate on a wide context representation, a high similarity score between terms indicates *topical similarity*.

Retrieving similar terms that are closer to *near-synonymy* can be obtained when using a restrictive and more precise context representation, which is achieved with term-based semantic models. For these models, relations between terms are considered, be it e.g. neighborhood relations or syntactic relations, as introduced in Section 2.4. Using these models, only similarities between terms can be computed.

To give an insight into the two different categories of models, we give examples for document similarities and term similarities computed on a small corpus. These exemplify the characteristics of different models and illustrate the advantages and disadvantages of each of the models.

3.1 Document-based Context Representations

In this section, we describe document-based models. First, we demonstrate the impact of term-document based VSMs. In order to reduce the contextual representation, we describe LSA and conclude the section by exemplifying the topic model LDA.

3.1.1 Document-based Vector Space Model (VSM)

The vector representation of terms in the so-called vector space model (VSM) is one of the most prominent model used in information retrieval (IR)[8]. VSMs based on a term-document representation can be used to compute similarities between documents and to retrieve relevant documents according to a query.

Instantiating our tuple-based formalism, specified in Section 2.5, we define a term as a *language element* and use the document as the *context features*. Let y_j be a document within the corpus Y containing $|Y|$ documents. Within the corpus we have $x_i \in |X|$ unique terms. The similarity computation is focused on frequencies. For a document-based VSM we represent the corpus as a $|X|\times|Y|$ matrix M with the frequencies of terms within the documents $M_{i,j} = |\langle x_i, y_j \rangle|$ as elements.

[8] According to (Manning et al., 2008, p. 1) the academic field of IR is *"finding information material (e.g. documents) of an unstructured nature (e.g. text) that satisfies an information need from within large collections"*.

We illustrate the various models by presenting similarities between documents and terms using a corpus of 10,000 randomly sampled documents from the New York Times (NYT) newspaper corpus.[9] This corpus consists of 192,633 types, which needs to be represented of a term-document vector space model with 1,9 billion values (number of types × documents). Using an Integer data type occupying 32 Bits, like in most modern compilers, and ignoring memory for headers and references, storing all counts requires 7.7GB of memory. This matrix is sparse as it contains 99.79% zero entries and 0.21% non-zero entries.

To exemplify the similarity between documents, we manually selected two documents from five different categories from the corpus: *baseball* (1-2), *basketball* (3-4), *ice hockey* (5-6), *beer* (7-8) and *politics* (9-10). We chose three sport categories, as these documents might be very similar due to their similar domain. The "beer" and "politics" categories are chosen, as they are different from each other and vary from the documents in the sport categories. Furthermore, we select the terms *pitcher*, *basket*, *NHL*, *beer* and *Bush* as representative term for each category. The relevance of the terms for each category is demonstrated in Table 3.1.

Doc. ID	pitcher	basket	NHL	beer	Bush
1	2	0	0	0	0
2	2	0	0	0	0
3	0	3	0	0	0
4	0	3	1	0	0
5	0	0	12	0	0
6	0	0	3	0	0
7	0	0	0	12	0
8	0	0	0	18	0
9	0	0	0	0	21
10	0	0	0	0	30

Table 3.1: This table presents the term frequencies for the 5 selected terms *pitcher*, *basket*, *NHL*, *beer* and *Bush* within 10 documents. The documents are classified in the following categories: *baseball* (1-2), *basketball* (3-4), *ice hockey* (5-6), *beer* (7-8) and *politics* (9-10).

There we show the frequencies of the terms for the selected documents. Most of the terms occur in only one category but the term *NHL* is also contained in one of the *basketball*-related document (4). The *beer* documents (8 and 9) describe different kinds of beer and are not about drinking beers. Otherwise, the term *pitcher* might not only refer to the sense of the baseball player but also to the jar and thus might have higher counts for the beer-related documents.

We use the cosine similarity for computing similarities between documents. For this, we can apply the Formula 2.13 in Section 2.8 and switch the order of the elements in the tuples. Computing the term similarity can be obtained with the same formula without any tuple modification. First, we present results for similarities computed between the selected documents

[9] We extracted the sentences from the AQUAINT corpus collection: `http://www.nist.gov/tac/data/data_desc.html#AQUAINT`.

relying on the frequency for the document-term based VSM. We present the similarities among all documents in Table 3.2.

	baseball		basketball		ice hockey		beer		politics	
Doc. ID	1	2	3	4	5	6	7	8	9	10
1	1.000	**0.867**	0.806	0.847	0.857	0.883	0.785	0.786	0.847	0.847
2	0.867	1.000	0.780	0.815	0.864	0.855	0.761	0.779	0.825	0.842
3	0.806	0.780	1.000	0.865	0.858	0.882	0.772	0.845	0.855	0.850
4	0.847	0.815	0.865	1.000	0.912	0.902	0.829	0.857	0.894	0.889
5	0.857	0.864	0.858	**0.912**	1.000	**0.925**	0.843	0.875	0.904	0.917
6	**0.883**	0.855	**0.882**	0.902	**0.925**	1.000	0.830	0.873	0.919	0.913
7	0.785	0.761	0.772	0.829	0.843	0.830	1.000	**0.879**	0.812	0.833
8	0.786	0.779	0.845	0.857	0.875	0.873	**0.879**	1.000	0.853	0.874
9	0.847	0.825	0.855	0.894	0.904	0.919	0.812	0.853	1.000	**0.945**
10	0.847	0.842	0.850	0.889	0.917	0.913	0.833	0.874	**0.945**	1.000

Table 3.2: Cosine similarities between documents based on the frequency of the term vectors of each document.

In each column, the highest scores, except for the ones on the main diagonal, are marked as bold. The values on the main diagonal denote the self-similarity for each document and thus always have a similarity score of 1.0. We observe that five out of the six sport related documents are most similar to the *ice hockey* documents. Using this document representation, the *politics*, *beer* and *ice hockey* documents display the highest similarities among each others. As we do not filter frequent and rare terms, the similarity computation is not only time-consuming, but also frequent non-content words deteriorate the quality of the similarity computations. To alleviate the influence of stopwords one can remove high and low frequent terms or weight terms using weighting function like the *tf-idf* (see Section 2.7). In a first experiment we remove the 100 most frequent terms and remove all terms with a frequency lower than 100.

This reduces the vocabulary size from 192,633 types to 32,882 types, which is 17.07%. Again, we compute the similarities and obtain document-based similarities as shown in Table 3.3. In this experiment, almost all topically related documents are most similar to each other. Only the Document 5 is most similar to a *basketball* document rather than the *ice hockey* document. However, the second highest similarity score is obtained for the "correct" *ice hockey* related Document 6. This better performance is affected by the pruning, as now the similarity is computed based on content words (nouns, verbs, adjectives, adverbs) rather than stopwords.

When using the *tf-idf* weighting as described in Section 2.7.2, we observe similarity scores between documents as illustrated in Table 3.4. All documents have the highest similarity score for the document within the same category. This time the similarity score for Document 5 is highest for the corresponding Document 6. But we also obtain a high score for the similarity to Document 4, which is similar to the filtered results presented in Table 3.3.

	baseball		basketball		ice hockey		beer		politics	
Doc. ID	1	2	3	4	5	6	7	8	9	10
1	1.000	**0.362**	0.107	0.168	0.125	0.131	0.039	0.082	0.086	0.085
2	**0.362**	1.000	0.105	0.145	0.159	0.157	0.049	0.083	0.072	0.096
3	0.107	0.105	1.000	**0.307**	0.155	0.161	0.054	0.140	0.100	0.058
4	0.168	0.145	**0.307**	1.000	**0.302**	0.250	0.057	0.143	0.117	0.097
5	0.125	0.159	0.155	0.302	1.000	**0.300**	0.093	0.120	0.155	0.159
6	0.131	0.157	0.161	0.250	0.300	1.000	0.068	0.133	0.133	0.135
7	0.039	0.049	0.054	0.057	0.093	0.068	1.000	**0.321**	0.058	0.058
8	0.082	0.083	0.140	0.143	0.120	0.133	**0.321**	1.000	0.093	0.083
9	0.086	0.072	0.100	0.117	0.155	0.133	0.058	0.093	1.000	**0.596**
10	0.085	0.096	0.058	0.097	0.159	0.135	0.058	0.083	**0.596**	1.000

Table 3.3: Cosine similarities between the documents computed on the term frequency vectors when filtering frequent and rare words.

	baseball		basketball		ice hockey		beer		politics	
Doc. ID	1	2	3	4	5	6	7	8	9	10
1	1.000	**0.350**	0.024	0.053	0.042	0.044	0.007	0.013	0.028	0.025
2	**0.350**	1.000	0.026	0.037	0.057	0.046	0.007	0.016	0.025	0.033
3	0.024	0.026	1.000	**0.151**	0.044	0.061	0.009	0.036	0.034	0.018
4	0.053	0.037	**0.151**	1.000	0.124	0.102	0.012	0.036	0.047	0.032
5	0.042	0.057	0.044	0.124	1.000	**0.143**	0.015	0.025	0.060	0.056
6	0.044	0.046	0.061	0.102	**0.143**	1.000	0.016	0.030	0.048	0.053
7	0.007	0.007	0.009	0.012	0.015	0.016	1.000	**0.196**	0.013	0.012
8	0.013	0.016	0.036	0.036	0.025	0.030	**0.196**	1.000	0.029	0.020
9	0.028	0.025	0.034	0.047	0.060	0.048	0.013	0.029	1.000	**0.409**
10	0.025	0.033	0.018	0.032	0.056	0.053	0.012	0.020	**0.409**	1.000

Table 3.4: Cosine similarities between the documents when weighting the terms according to tf-idf.

Using a document term representation we also can also compute similarities between words using the cosine similarity as described in Equation 2.13 in Section 2.8. We show the ten most similar terms for *pitcher*, *basket*, *NHL*, *beer* and *Bush* in Table 3.5. Here, we apply document co-occurrences as context representations.

pitcher	1.00	basket	1.00	NHL	1.00	beer	1.00	Bush	1.00
innings	0.458	16-foot	0.393	Bettman	0.482	aperitif	0.582	George	0.482
pitchers	0.427	42-40	0.393	43.7	0.441	Ardennes	0.582	campaign	0.469
pitches	0.408	67-65	0.393	59.4	0.441	Biere	0.582	presidential	0.453
pitch	0.406	68-67	0.393	hard-stance	0.441	Blatz	0.582	W.	0.445
fastball	0.399	Adubato	0.393	Kraus	0.441	Chouffe	0.582	McCain	0.444
pitching	0.396	Weatherspoon	0.391	NHLPA	0.441	corner-tavern	0.582	Gov.	0.434
pitched	0.363	16,285	0.371	Payout	0.441	D'Achouffe	0.582	Republican	0.425
mound	0.354	65-63	0.371	Payrolls	0.441	estery	0.582	governor	0.425
batters	0.351	17-2	0.324	receipt-driven	0.441	Ghassemian	0.582	Governor	0.382

Table 3.5: Compute cosine similarities between terms based on document occurrences without filtering.

We obtain topical similar terms for the term *pitcher*, which mostly share the stem *pitch*. Whereas for the term *Bush* we observe many topically related terms, the term *beer* is most similar to brand names and breweries. For the terms *NHL* and *basket* we observe some player names and numeric values, which are scoring results or playing times. Data analysis reveals that most of these terms are rare and only occur in one document. As the denominator is always one for the rare term, the score is still quite high if the original term (e.g. *beer*) often occurs in the same document. This seems to be an issue for the cosine similarity and is related to the curse of dimensionality (Bellman, 1957).[10]

pitcher	1.00	basket	1.00	NHL	1.00	beer	1.00	Bush	1.00
innings	0.458	Weatherspoon	0.391	Bettman	0.482	ale	0.536	George	0.482
pitchers	0.427	jumper	0.3	hockey	0.405	beers	0.498	campaign	0.469
pitches	0.408	points	0.293	defenseman	0.396	Budweiser	0.439	presidential	0.453
pitch	0.406	3-pointer	0.285	Hockey	0.352	brewery	0.396	W.	0.445
fastball	0.399	fouled	0.284	Hayward	0.346	Oktoberfest	0.391	McCain	0.444
pitching	0.396	rebounds	0.266	envious	0.333	Soleil	0.267	Gov.	0.434
pitched	0.363	lane	0.264	goalie	0.302	tasting	0.253	Republican	0.425
mound	0.354	Comets	0.256	puck	0.301	Draft	0.246	governor	0.425
batters	0.351	halftime	0.245	Coyotes	0.293	Murree	0.226	Governor	0.382

Table 3.6: Cosine similarities between terms based on document co-occurrence with term filtering.

The sparsity issues are resolved when filtering frequent and rare terms as shown in Table 3.6. We observe similar results as before for the terms *pitcher* and *Bush*, which already have been most similar to frequent terms in the previous computation. More closely related terms

[10]Like many other distance functions, the cosine similarity additionally has problems with sparse vectors, which is explained in more detail in Appendix A.

are obtained for the remaining terms. Whereas for the *beer*-related terms in the previous Table 3.5 we observed very infrequent terms, we now achieve more reasonable results e.g. *ale*, *beers* or *Budweiser*, which are more related to the term *beer*. Using the tf-idf weighting for computing similarities between terms is equivalent to using the term frequency, as the inverse document frequency is just a factor, which applies to all frequencies of each term, and can be canceled from the cosine similarity.

The document-term-based VSM is a quite precise method when applying filtering and using normalization. However, the main problem is the size of the matrix and its sparsity. Whereas the information of the sparse matrix can be stored efficiently (e.g. Press et al., 2007, p. 75), still the similarity calculations become time-consuming when using larger corpora especially as the similarity computations have to be performed between all terms or documents. A further drawback of using frequency-based approaches is that similarities can only be computed when there is an overlap of the contextual representation. Thus, we cannot compute the similarity between two terms if they do not co-occur within the same document.

3.1.2 Latent Semantic Analysis (LSA)

Pruning the document-term VSM does only partially solve the sparsity and memory issues. In this section, we focus on a method, which results to a denser and smaller context representation. With LSA (Deerwester et al., 1990) a principle was introduced, which reduces the dimensions of the vector space and captures semantics properties. LSA itself is based not a new method for dimensionality reduction but is an adaptation of the singular value decomposition (SVD) to natural language. This model uses the document-term based VSM matrix M as input (see Section 3.1.1). Applying the SVD, the matrix is decomposed into $M = UDV^T$ (see Figure 3.1) with U representing the terms, the singular values D and the documents V. To reduce the dimensions, the decomposed elements are used with the highest singular values

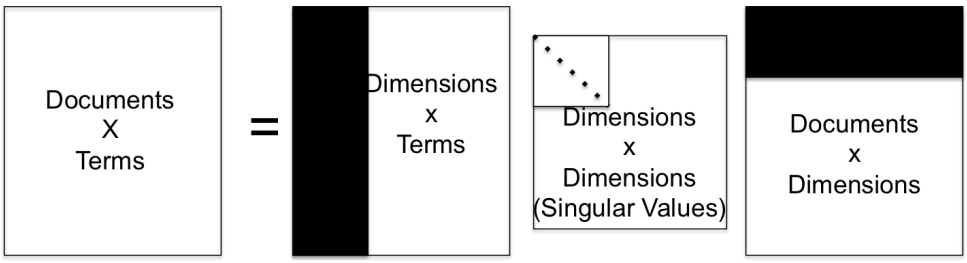


Figure 3.1: Visualization of the components after applying LSA on a document term matrix.

(indicated with the black bars in Figure 3.1). The singular values are equal to the square root of the Eigenvalues of the matrix M. Using this method alleviates the sparsity issues and allows representing terms and documents with a denser representation. The main goal of the SVD is to extract the n most important representations from the document-term matrix, resulting

in the reduced matrices U^n, D^n and V^n. Multiplying the reduced matrices again approximates the document-term matrix: $M \approx U^n D^n V^n$.

In line with the examples in the previous section, we show results for the document similarity by using the reduced document matrices. In Figure 3.2 we demonstrate the first two dimensions, which have the highest singular values, of V^n multiplied with the according singular values D^n, serving as weights for the dimensions.

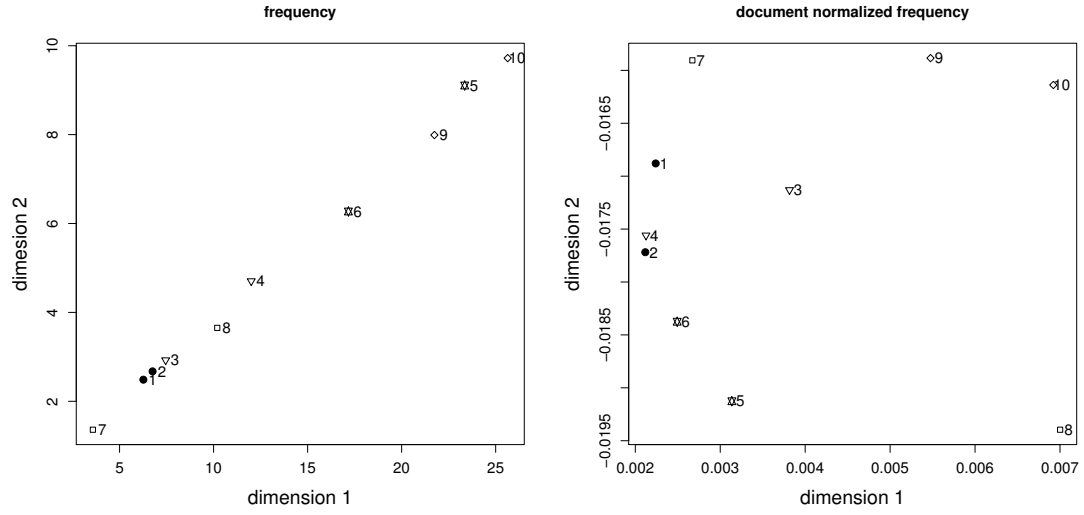


Figure 3.2: This figure shows the document representation based on the components with the highest singular values of LSA.

The left graph in Figure 3.2 demonstrates the highest two components when computed based on the raw frequency counts of the document. The right graph shows the first two dimensions when the term-frequency is normalized by document length. Using raw frequencies, the SVD performs poorly, as the first two dimensions seem to be highly correlated for the selected documents.[11] The plot on the right side illustrates results that are more reasonable and makes better usage of the vector space. Whereas all the sport related documents (1-6) are grouped together in the left corner, the *politic* news articles are in the top right corner (9,10). We can also again use the reduced dimensions to compute term similarity. Only the beer-related documents (7,8) are not represented together based on the first two dimensions.

When computing the similarities among the ten documents, we considered different number of dimensions. We show the similarities considering the top ten dimensions in Table 3.7, for the highest 100 dimensions in Table 3.8 and for 1000 dimensions in Table 3.9. Considering only the first ten dimensions does not look as promising as the results shown in Figure 3.1. Only the political documents can be aligned. But also the sport documents are highly similar to each other. A high alignment between the topically similar documents is already observed

[11] We achieve a Pearson correlation of 0.998 between the first two dimensions based for our ten documents.

	baseball		basketball		ice hockey		beer		politics	
Doc. ID	1	2	3	4	5	6	7	8	9	10
1	1.000	0.996	**0.977**	0.994	0.956	0.933	0.933	0.915	0.718	0.727
2	**0.996**	1.000	0.976	**0.997**	0.969	0.945	0.942	0.918	0.729	0.735
3	0.977	0.976	1.000	0.973	0.959	0.939	0.915	0.887	0.739	0.749
4	0.994	**0.997**	0.973	1.000	0.949	0.929	0.921	0.909	0.682	0.691
5	0.956	0.969	0.959	0.949	1.000	**0.970**	**0.981**	0.938	0.867	0.867
6	0.933	0.945	0.939	0.929	0.970	1.000	0.949	0.904	0.841	0.836
7	0.933	0.942	0.915	0.921	**0.981**	0.949	1.000	**0.974**	0.899	0.904
8	0.915	0.918	0.887	0.909	0.938	0.904	0.974	1.000	0.849	0.866
9	0.718	0.729	0.739	0.682	0.867	0.841	0.899	0.849	1.000	**0.997**
10	0.727	0.735	0.749	0.691	0.867	0.836	0.904	0.866	**0.997**	1.000

Table 3.7: Similarities between the documents using the first 10 dimensions of an LSA model.

when using the 100 dimensions with the highest singular values, except for the *beer*-related Document 7. So far, the best alignment was achieved for the pruned VSM, but here the number of contexts used for LSA computation relies on a much denser representation. Using a term-based VSM without filtering represents each document by a vector of 192,633 types, which results to higher computational efforts.

	baseball		basketball		ice hockey		beer		politics	
Doc. ID	1	2	3	4	5	6	7	8	9	10
1	1.000	**0.771**	0.466	0.509	0.382	0.497	0.332	0.373	0.199	0.215
2	**0.771**	1.000	0.390	0.505	0.571	0.661	0.517	0.491	0.226	0.255
3	0.466	0.390	1.000	**0.801**	0.480	0.544	0.358	0.436	0.232	0.178
4	0.509	0.505	**0.801**	1.000	0.644	0.680	0.389	0.519	0.202	0.171
5	0.382	0.571	0.480	0.644	1.000	**0.806**	**0.721**	0.501	0.348	0.311
6	0.497	0.661	0.544	0.680	**0.806**	1.000	0.688	0.589	0.350	0.317
7	0.332	0.517	0.358	0.389	0.721	0.688	1.000	**0.624**	0.406	0.390
8	0.373	0.491	0.436	0.519	0.501	0.589	0.624	1.000	0.247	0.236
9	0.199	0.226	0.232	0.202	0.348	0.350	0.406	0.247	1.000	**0.945**
10	0.215	0.255	0.178	0.171	0.311	0.317	0.390	0.236	**0.945**	1.000

Table 3.8: Similarities between the documents represented by their 100 most relevant dimensions of the LSA model.

As we observe from Table 3.9, using the top 1000 dimensions leads to the highest similarities between the two *beer* documents. Thus, considering these context representations the best alignment between similar documents is achieved (cf. results in Table 3.3). Similar results are obtained when applying all context representations, but require more computational efforts.

In the next experiment, we examine the similarities among terms using the vector representation. We selected different number of dimensions and rely on the pruned vocabulary.

	baseball		basketball		ice hockey		beer		politics	
Doc. ID	1	2	3	4	5	6	7	8	9	10
1	1.000	**0.534**	0.170	0.265	0.196	0.236	0.088	0.151	0.140	0.120
2	**0.534**	1.000	0.169	0.240	0.249	0.269	0.115	0.176	0.111	0.136
3	0.170	0.169	1.000	**0.457**	0.226	0.250	0.105	0.263	0.145	0.081
4	0.265	0.240	**0.457**	1.000	0.421	0.409	0.109	0.258	0.167	0.119
5	0.196	0.249	0.226	0.421	1.000	**0.471**	0.178	0.209	0.223	0.213
6	0.236	0.269	0.250	0.409	**0.471**	1.000	0.136	0.238	0.225	0.204
7	0.088	0.115	0.105	0.109	0.178	0.136	1.000	**0.429**	0.114	0.109
8	0.151	0.176	0.263	0.258	0.209	0.238	**0.429**	1.000	0.151	0.126
9	0.140	0.111	0.145	0.167	0.223	0.225	0.114	0.151	1.000	**0.749**
10	0.120	0.136	0.081	0.119	0.213	0.204	0.109	0.126	**0.749**	1.000

Table 3.9: Similarities between document representations considering the 1000 most relevant dimensions according to the singular values of the LSA model.

When using the first ten dimensions, we find many terms, which have high similarity scores as shown in Table 3.10. This is attributed to rounding the floating number to two digits. Again, we observe similarities to rare terms, which are topically related. Most similar terms for *NHL* are upper case terms. Additionally, the topical relatedness for the similar terms for *beer* is not

pitcher	1.00	basket	1.00	NHL	1.00	beer	1.00	Bush	1.00
opener	1.00	championship	1.00	ESPN	0.96	chair	1.00	vote	1.00
Tigers	1.00	offensive	1.00	HERE	0.95	landing	1.00	election	1.00
coach	1.00	seasons	1.00	through	0.94	looking	1.00	executive	1.00
pitch	1.00	play	1.00	NBA	0.94	heart	1.00	constitutional	1.00
pitchers	1.00	coaching	1.00	NASCAR	0.94	stolen	1.00	cash	1.00
game	1.00	Arena	1.00	D.C.	0.91	surface	1.00	employees	1.00
All-Star	1.00	hitting	1.00	Ga.	0.91	native	1.00	policy	1.00
pitched	1.00	seconds	1.00	today	0.91	courses	1.00	bank	1.00
Piazza	1.00	score	1.00	fix	0.90	ones	1.00	initiative	1.00

Table 3.10: Similarities between the selected terms using the highest 10 dimensions for the LSA-based document-term matrix.

as precise as when using the *VSM*-based approach. Thus, we increase the dimensions to 1000 (see Table 3.11), which performs best when comparing documents.

This improves the similarity between the terms, as we obtain more topically related similarities for the term *beer* as when using only 10 dimensions. Using 1000 dimensions does not resolve this issue.

Due to the reduction of the word vectors, similarity computation can be performed more efficiently. When using 1000 dimensions, we observe the highest similarity scores between the documents of the same topics. The similarity between terms seems reasonable but achieves inferior results for rare terms.

pitcher	1.00	basket	1.00	NHL	1.00	beer	1.00	Bush	1.000
pitchers	0.54	throws	0.44	goalie	0.48	juice	0.42	advisers	0.168
pitched	0.52	rebounds	0.42	NBA	0.46	festival	0.40	Governor	0.155
clubhouse	0.51	foul	0.35	defenseman	0.42	sauce	0.35	campaigning	0.108
pitch	0.50	Liberty	0.31	Ducks	0.38	tap	0.33	primaries	0.102
pitches	0.49	points	0.29	UPDATE1	0.34	celebration	0.30	rivals	0.097
fastball	0.44	half	0.29	Anaheim	0.32	traditional	0.28	polls	0.092
ERA	0.41	shots	0.27	stick	0.29	typical	0.28	nomination	0.077
pitching	0.41	seconds	0.26	hockey	0.29	define	0.27	Cheney	0.076
mound	0.41	assists	0.24	SPORTS	0.24	anything	0.26	debates	0.071

Table 3.11: Similarities between the selected terms using the first 100 dimensions of the LSA model.

Whereas LSA can be easily applied to the VSM, the SVD computation is slow when not using optimized implementations (e.g. Holmes et al. (2007)). Like most clustering algorithms the number of dimensions to reduce needs to be detected. Furthermore, LSA is not capable to handle terms with more than one meaning and expects data following the normal distribution (see Manning and Schütze, 1999, p. 565). This is not given when working with words, as they follow a power-law distribution as described in Section 2.6. Whereas LSA captures semantic information within the reduced dimensions, each dimension itself does not contain any semantic information. For example extracting the highest ranked words of a dimension does not result to similar terms. In order to capture semantic information within the different dimensions, topic model can be applied, like LDA, which we describe in the next section.

3.1.3 Latent Dirichlet Allocation (LDA)

Hofmann (1999) introduced one of the first topic model called pLSA. Here we apply the topic model called Latent Dirichlet Allocation (LDA), which is the first complete generative topic model and was introduced by Blei et al. (2003). LDA assumes a document to consist of a distribution of topics. A topic is considered as a list of words with probabilities indicating their belonging to a certain topic. Using LDA each word is contained in each topic with different probabilities. Thus, each term itself follows topic distribution. The topic-term and topic-document distributions are estimated using a generative process, which can be explained based on the graphical model in Figure 3.3:

First, for each document $d \in D$ a topic distribution ϕ with prior β is sampled. Then, for each document, N terms (w) are randomly chosen, following the previously sampled topic distribution. The estimation of the posterior probability is approximated as its directly computation is intractable. For the estimation, Collapsed Gibbs sampling is often used (Press et al., 2007, p. 827-828), which is both applied for training a model and for predicting the topic distributions for unseen documents. During the Gibbs sampling, each term is assigned with a topic identifier, which is sampled based on the distribution of the term-topic and the topic-document

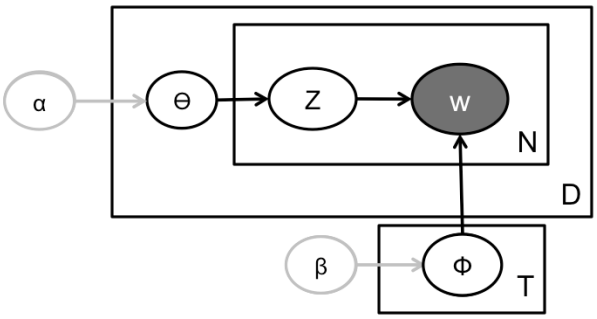


Figure 3.3: This figure illustrates the plate notation for the generative process of LDA.

distribution. Note that the so-called inference procedure, which assigns topics to unseen documents, marks the difference between LDA and earlier dimensionality reduction techniques such as LSA. LDA has three parameters that need to be specified: The number of topics (T) and two hyperparameters α and β. Whereas α specifies the skewness of the document topic distribution, β is used as parameter for the distribution of the term-topic distribution. The topic-document distribution is approximated using α and the word topic distribution is selected according to β[12]. A detailed sweep over the parameters of LDA and their influence in a text segmentation application is presented in Chapter 4.

An insight is given into LDA using the 10,000 documents from the New York Times. We train the model based on $T = 100$ topics.[13] No term filtering is performed for the computation and no further normalization (cf. LSA in the previous section) is necessary. After the training each topic is represented as a list of all words with probabilities, as the topic hyperparameter β, which also serves as smoothing parameter, avoids zero probabilities. Examining the top 15 most probable words for each topic in Table 3.12, we observe many stopwords, which do not seem to be topic relevant. Stopwords are frequent among all documents and co-occur with all words. Thus, they also receive the highest probabilities for each topic.

Whereas topic models have been modified to cluster stopwords into its own topic (Wallach et al., 2009), these approaches are much slower. Including stopwords into the training increases the runtime and number of iterations needed for the training and adds some noise into the topics. Mainly they are removed to improve the interpretability of topics for humans. However, topically coherent words are still grouped together when including stopwords. The stopwords can be seen as a snow layer, which only hides the structure. When using LDA for text segmentation as described in Chapter 4 we gain only slight improvements when filtering stopwords (see Section 4.4.3). If we filter the 100 most frequent terms from each topic and examine the same topics again (see Table 3.13), the topic representations can more concisely

[12]Whereas α and β could be specified asymmetric for each document respectively word (more details are described by Wallach et al. (2009)), we concentrate on symmetric priors, leading to a constant value for both α and β.

[13]We use standard parameters for the hyperparameters with $\alpha = 20/T$ and $\beta = 0.01$.

Topic	1th:	Topic	22th:	Topic	76th:	Topic	78th:
the	0.0664	the	0.0698	,	0.0891	the	0.0551
.	0.0620	,	0.0568	.	0.0479	,	0.0501
,	0.0540	.	0.0502	the	0.0456	.	0.0428
and	0.0318	in	0.0325	and	0.0348	to	0.0345
a	0.0264	and	0.0274	a	0.0327	and	0.0326
in	0.0263	a	0.0260	of	0.0289	of	0.0311
to	0.0177	to	0.0168	for	0.0179	in	0.0271
of	0.0140	of	0.0133	in	0.0174	United	0.0255
game	0.0138	for	0.0109	food	0.0153	States	0.0221
's	0.0132	his	0.0107	to	0.0131	a	0.0168
with	0.0101	with	0.0103	with	0.0104	China	0.0153
The	0.0098	The	0.0090	The	0.0090	American	0.0135
team	0.0097	's	0.0084	from	0.0085	U.S.	0.0113
for	0.0086	game	0.0083	's	0.0081	with	0.0113

Table 3.12: In this table we show four topics from LDA computed on 10,000 NYT documents without applying any word filtering.

be interpreted as in the previous representation. Whereas topic 1 is about *basketball*, topic 22 focuses on *baseball*. Topic 76 is about food and eating and the 78th topic contains country names as well as politically related terms and could be interpreted as *foreign affairs*.

For sampling the topics of LDA, we use Gibbs sampling, which is a randomized algorithm. Thus, the topic identifiers change with every run. In addition to the estimation of a topic model, LDA features a mechanism, called inference method, which allows estimating the topic distribution for previously unseen documents. For this, it assigns each word with a topic identifier based on the term-topic distribution of an already computed model and considers the topic-document distribution of the new document. Using these assignments, we can also compute the topic distribution of the unseen data. For the example sentence in Figure 3.4 we observe that most of the terms are assigned with the computer topic 93. The mechanism of topic identifier assignments is used for Text Segmentation (TS) in Chapter 4.

The	mouse	likes	to	eat	cheese	.
90	9	40	2	77	77	77

Figure 3.4: Example sentence after it has been assigned with topic identifiers by the LDA model computed on 10,000 documents.

Based on the topic representation of a word, we can also compute similarities between words based on the cosine similarity, as shown in Table 3.14. We observe that except for the term *beer* most terms are topically related. For *basket* we find as most similar terms trainer names (*Rick Majerus*), stadium names (*Autzen Zoo*) or player names (*Muggsy Bogues*, *Antawn Jamison*). Similar topic-related names are observed for the terms *NHL*, *Bush* and *pitcher*.

Topic	1th:	Topic	22th:	Topic	76th:	Topic	78th:
game	0.0138	game	0.0083	food	0.0153	United	0.0255
team	0.0097	Yankees	0.0058	wine	0.0068	States	0.0221
points	0.0078	season	0.0057	restaurant	0.0048	China	0.0153
coach	0.0075	home	0.0056	eat	0.0042	American	0.0135
play	0.0070	hit	0.0054	farmers	0.0038	U.S.	0.0113
season	0.0070	games	0.0053	eating	0.0033	Chinese	0.0093
games	0.0060	runs	0.0053	meat	0.0031	countries	0.0078
basketball	0.0058	baseball	0.0047	milk	0.0030	trade	0.0073
State	0.0055	Dodgers	0.0046	good	0.0029	Clinton	0.0068
Lakers	0.0044	run	0.0043	products	0.0028	foreign	0.0066
against	0.0043	against	0.0041	foods	0.0027	officials	0.0064
NBA	0.0043	Mets	0.0041	beer	0.0027	administration	0.0063
guard	0.0043	inning	0.0039	farm	0.0026	government	0.0057
players	0.0040	Sox	0.0038	made	0.0022	world	0.0054

Table 3.13: We show four topics computed with LDA and additionally filtered the 100 most frequent terms.

Whereas the similar terms for *Bush* are similar to the results achieved with the other methods, the similarities for the other terms vary. The most similar terms for *beer* are not beer-related, but still related to drinks and food. We observe many equal similarity scores, which

pitcher	1.00	basket	1.00	NHL	1.00	beer	1.00	Bush	1.00
pitching	1.00	Autzen	1.00	Vault	0.99	bottle	1.00	Gore	1.00
Mariners	1.00	Alford	0.99	Bayern	0.99	bottles	0.99	McCain	1.00
innings	1.00	perimeter	0.99	Axelsson	0.99	eats	0.99	Dole	1.00
McGwire	1.00	Spartan	0.99	midfielders	0.99	eat	0.99	D'Amato	1.00
baseman	1.00	Majerus	0.97	Matthaeus	0.99	grocery	0.99	Cheney	1.00
Griffey	1.00	inbounded	0.97	Aubin	0.99	flocks	0.99	Kasich	1.00
homer	1.00	season-low	0.97	Brenden	0.99	snack	0.99	pollster	1.00
Sosa	1.00	Muggsy	0.97	Sigi	0.99	drink	0.99	Mauro	1.00
Cubs	1.00	Antawn	0.97	Impe	0.99	hamburger	0.99	conservatism	1.00

Table 3.14: Most similar words for the example terms using the topic word vectors computed with LDA.

is attributed to the rounding to two digits. LDA can also be used to compute document similarities using the topic distribution of each document. The results are presented in Table 3.15. Here we observe a matching between the topically similar documents.

In contrast to the LSA similarities (see Table 3.8), we observe good results with only 100 dimensions. This might be attributed, as LDA is better suited to fit to the data. With LDA a method is described, which is able to extract topical structure from text. Using 100 dimensions already allows us to obtain a similarity matching between topically similar documents. The dimensions of this model capture topical properties, as ranking terms according to its topic-term

	baseball		basketball		ice hockey		beer		politics	
Doc. ID	1	2	3	4	5	6	7	8	9	10
1	1.000	**0.940**	0.041	0.056	0.085	0.290	0.083	0.040	0.044	0.06
2	**0.940**	1.000	0.064	0.090	0.144	0.283	0.129	0.074	0.053	0.09
3	0.041	0.064	1.000	**0.880**	0.118	0.199	0.076	0.083	0.094	0.08
4	0.056	0.090	**0.880**	1.000	0.237	0.276	0.144	0.111	0.093	0.08
5	0.085	0.144	0.118	0.237	1.000	**0.658**	0.147	0.097	0.117	0.11
6	0.290	0.283	0.199	0.276	**0.658**	1.000	0.162	0.164	0.110	0.12
7	0.083	0.129	0.076	0.144	0.147	0.162	1.000	**0.790**	0.122	0.12
8	0.040	0.074	0.083	0.111	0.097	0.164	**0.790**	1.000	0.083	0.09
9	0.044	0.053	0.094	0.093	0.117	0.110	0.122	0.083	1.000	**0.932**
10	0.068	0.099	0.086	0.089	0.117	0.125	0.124	0.095	**0.932**	1.00

Table 3.15: Cosine similarities between the ten example documents based on the topic distributions retrieved with LDA.

probability results to topically related words. Furthermore, terms can have high probabilities in more than one topic, which allows a term to have more than one meaning.

3.2 Term-based Context Representations

As *term-based context representations* we classify all models, which do not rely on the document. These models represent a *language element* (e.g. term) by *context features* as defined by syntagmatic relations. This could vary from direct neighboring terms up to complex context features, which could be terms in conjunction with syntactic relationships or/and POS tags. It is obvious that using context, which is closer to the word, also results to more synonymy paradigmatic relations then using solely document information as context. Instead of computing similarities based on the context features shared by two *language elements*, we first show results obtained with co-occurrence models. These models figure out *language elements*, which significantly co-occur with another *language element*.

3.2.1 Co-occurrences

Another approach to extract related *language elements* is the usage of co-occurrence relations, which is also called *first order co-occurrences*.[14] This approach is based on a VSM, counting *language elements* X and their co-occurring *language elements* as *context features*, leading to a $|X| \times |Y|$ matrix. In order to reduce computational efforts, the co-occurring terms considered are restricted within a window of the n previous and following words. We present examples

[14]The first order co-occurrence can be used as context representations for computing similarities. These are then called second order co-occurrences

computed on the sampled newspaper corpus and use the software TinyCC[15] for computing the co-occurrences. This software can be operated both on direct neighboring words or all words within a sentence. To measure whether two words do not co-occur by chance, a significance measurement is computed between the two terms. TinyCC applies the Log-Likelihood (LL) (Dunning, 1993) significance measure for ranking two terms according to their relatedness. In Table 3.16 we show the co-occurring terms for the five terms with the highest significance score based on a neighborhood relationship. For the term *basket*, we do not observe terms,

pitcher		basket		NHL		beer		Bush	
Kevin	54.86	.	52.32	’	62.35	bottles	68.67	’	993.40
who	50.83	,	50.00	PLAYER	52.66	bottle	60.25	campaign	834.47
in	45.51	tied	20.32	GREAT	48.28	and	50.85	has	632.05
Randy	41.78	case	19.87	DRAFTS	42.74	,	45.23	said	526.88
,	41.29	came	19.47	AVERAGE	36.01	maker	31.41	,	191.92
Aaron	30.97	security	15.94	players	35.68	drinkers	28.89	is	181.78
Rolando	30.39	cases	15.66	FINES	35.10	.	27.22	administration	167.89
Tom	28.22	of	9.88	debut	30.55	industry	26.95	and	147.95
.	26.13	for	8.70	season	29.37	joints	26.67	spokesman	100.76
Nolan	22.93	with	7.72	history	27.74	cans	25.47	told	84.49

Table 3.16: The ten most significant co-occurrences for five terms without any filtering based on a direct neighborhood relationship ($n = 1$).

which are mostly related to the sport. The term *security* for example appears due to the term *basket D security*, which is a hybrid model from banks and insurance companies. The highest ranked co-occurring terms for *beer* and *Bush* are topically related. Punctuation signs are artifacts of the Log-Likelihood (LL) measure used, which are introduced by their high frequency. The related terms for *pitcher* are mostly similar to names of pitcher player.

Next, we compute the co-occurrences again but we filter the 100 most frequent terms in order to avoid stopwords and remove words with a frequency below 100 to remove the long tail in the Zipf law distribution (see Table 3.17). Whereas punctuation signs and stopwords vanish, still the co-occurring terms for *beer*, *Bush* and *NHL* are only slightly related to the terms to their highest co-occurring terms. For the terms *pitcher* we again observe many of names and for *basket* no related term seems to be obvious.

In the next experiment, we present results for co-occurrences when considering all terms within the sentence as *context feature*. Additionally, we filter again the 100 frequent terms and terms with a frequency below 100 (see Table 3.18). In contrast to the previous results, the highest ranked terms are more topically related to the searched terms. Here also the co-occurring terms with *beer* are related words. This is the case as the term *beer* is using in enumerations. The co-occurring terms for the term *basket* are only slightly topically related. Additionally, we observe different senses of basket namely the basket used in basketball (see the terms

[15] TinyCC 2.0 was developed by Chris Biemann and can be downloaded at http://wortschatz.uni-leipzig.de/~cbiemann/software/TinyCC2.html.

pitcher		basket		NHL		beer		Bush	
mound	61.26	communications	16.76	players	50.25	bottle	63.64	campaign	962.09
Kevin	53.51	tied	16.69	history	27.94	maker	25.96	Texas	450.55
Randy	34.52	case	14.53	season	26.79	bar	23.02	McCain	271.47
Mike	29.83	came	14.14	debut	25.11	industry	19.82	advisers	144.61
Pedro	26.01	security	12.34	note	22.20	sales	14.51	administration	133.01
Aaron	25.52	cases	12.07	U	21.77	ice	13.49	Republican	119.00
Boston	22.00	final	10.53	draft	21.58	three	12.29	aides	115.84
Jose	21.31	name	9.91	feature	20.15	spot	12.23	Gore	105.65
baseball	21.30	half	9.73	team	19.75	course	8.00	plan	103.47
win	21.17	-		goal	17.57	thought	7.53	spokesman	80.94

Table 3.17: List of the ten most significant co-occurrences based on a direct neighborhood relationship for five terms with filtering frequent and rare terms.

pitcher		basket		NHL		beer		Bush	
Yankees	212.98	eggs	53.49	season	121.17	bottle	54.44	George	5228.02
baseball	176.73	ball	50.48	SPORTS	91.99	cold	41.33	W	4672.31
innings	144.25	left	50.39	Stars	66.98	drinking	38.10	McCain	2161.90
Braves	136.91	seconds	47.11	Editors	66.96	drink	36.56	Gore	1827.93
Sox	121.48	shot	43.97	BUDGET	66.29	candy	35.59	campaign	1691.46
Rangers	115.60	put	28.23	BE	66.13	bar	35.20	Republican	1352.34
Dodgers	107.48	Liberty	26.92	ON	65.07	commercials	30.67	Texas	1172.87
Martinez	101.62	half	26.01	teams	64.43	alcohol	21.66	presidential	863.66
relief	101.18	front	25.63	Cup	61.29	cups	21.09	.	831.66
starting	92.65	drive	23.27	IS	60.03	wine	20.79	governor	725.95

Table 3.18: List of the ten most significant sentence co-occurrences for five terms with filtering frequent and rare terms.

ball, *shot*) and basket as a bin (e.g. *eggs*). Whereas the method does not rely on similarity computations, the co-occurrences we achieve with a sentence wide window are related. The computational effort of these models is minor in comparison to compute similarities between *language elements* relying on all *context features* they have in common, which we inspect next.

3.2.2 Term-based Vector Space Model

Whereas in the previous section we have just looked at co-occurring *language elements*, here we compute similarities between *language elements* (terms) X using the left and right neighboring term as *context feature* Y. This leads to a matrix M of $|X|\times|Y|$ dimensions. The similarity is again computed using the cosine similarity as introduced in Equation 2.13.

As the context of a term is less restrictive than using the document the terms occur, also the similarities are of lower quality as can be observed in Table 3.19. Whereas using document-based methods similarities are often within the same domain, we now observe e.g. for the term *NHL* a mixture of different sports. For the term *basket*, a relation between the similar terms is hardly observed. In addition, we observe that many words have different POS than

the words they are similar to. The remaining most similar words for the selected terms are topically related.

pitcher	1.00	basket	1.00	NHL	1.00	beer	1.00	Bush	1.00
pitchers	0.49	oath	0.35	NBA	0.55	water	0.33	Chang	0.88
lineup	0.44	consideration	0.35	NFL	0.55	eat	0.30	George	0.78
rotation	0.43	pressure	0.34	baseball	0.44	better	0.29	Richard	0.28
quarterback	0.36	fire	0.33	league	0.38	get	0.29	Howard	0.27
shortstop	0.35	way	0.31	U	0.35	worse	0.28	Pataki	0.25
starter	0.34	circumstances	0.30	football	0.33	because	0.28	Gov.	0.25
catcher	0.34	lying	0.30	UCLA	0.32	offer	0.28	Va.	0.23
player	0.32	scrutiny	0.29	hockey	0.32	still	0.27	trail	0.23
team	0.31	trees	0.29	basketball	0.31	work	0.27	contributions	0.23

Table 3.19: Similarities between terms, considering the left and right neighboring word as context.

In the next experiment, we consider all co-occurring terms within a sentence as *context feature*. Providing more context information for computing similarities between terms, the similar *language elements* for *basket* are more related than in the previous experiment (see Table 3.20). Nevertheless, using a larger context windows, results to similarities to terms of

pitcher	1.00	basket	1.00	NHL	1.00	beer	1.00	Bush	1.00
Rangers	0.72	shot	0.64	S	0.99	might	0.56	Gov.	0.85
pitchers	0.71	ball	0.63	NBA	0.99	actually	0.55	presidential	0.82
catcher	0.69	left	0.61	SPORTS	0.99	where	0.54	campaign	0.77
pitching	0.69	off	0.60	ESPN	0.97	little	0.54	George	0.77
season	0.69	behind	0.59	Sports	0.96	only	0.54	Republican	0.77
starter	0.69	seconds	0.58	USC	0.95	still	0.54	W.	0.77
baseball	0.68	back	0.58	NFL	0.92	enough	0.54	primary	0.76
Yankees	0.68	down	0.58	BUDGET	0.92	way	0.53	rival	0.73
Dodgers	0.68	lead	0.57	A&M	0.91	idea	0.53	governor	0.73

Table 3.20: Similarities between words considering all words within a sentence as context in a VSM.

different POS and topically related terms. The terms with the highest similarity score for *beer* are not related, but for *Bush*, *NHL* and *pitcher* the most similar terms are more topically related. Using this VSM-based similarity computation is a rather simple method for computing similarities, which has been extended to more sophisticated algorithms, leading to distributional thesauri (DTs) and is introduced in the next section.

3.2.3 Distributional Thesauri

According to Berry (1962) "A thesaurus is a book of words grouped by ideas. With a dictionary you know the word and need to find its meaning. With a thesaurus you have the idea

and you want to find the right words to express it - so all the words and phrases are classified according to ideas". We refer to a thesaurus as a distributional thesaurus (DT) if it was built automatically. For this, methods are required that compute similarities between *language elements* using good context representations. Using the previous methods, we observe topically related *language elements*. Here, we aim to model similarities, which are closer to near-synonymy, using more complex context representations. The term DT was first introduced by Lin (1997, 1998). This approach uses dependency relations for computing similarities between terms. Whereas the document-based VSM targets to give the ability to compute similarities between different *language elements* (e.g. terms or documents), the algorithms applied to compute DTs target at retrieving similarities of higher quality and thus concentrate on terms. Most approaches consider different context features, introduce different weighting functions for *language elements* and *context features* and apply different similarity functions.

Lin (1997) uses syntactic dependency relations between two terms, which form the contextual representation. We present his approach based on our tuple-based representation (see Section 2.5). Considering two terms t_i and t_j connected with a syntactic dependency relations r_k results to the tuple $\langle t_i, (t_j, r_k)\rangle$ with the *language element* t_i and *context feature* (t_j, r_k). Lin (1997) uses the same relations for word sense disambiguation, as it can be assumed that words with exactly the same dependency relations should have a similar meaning. The work presented by Lin (1998) goes one step further and computes co-called second-order representation, which characterize similarities between terms using the context representation. According to de Saussure (1959), these form the *paradigmatic* relations. Most computations for second-order relations between terms rely on two steps. First, an informativeness score is computed between the *language elements* and *context feature*. This score captures informativeness of the *language element-context feature* representation. In order to compute second-order relationship between two *language elements* t_i and t_j, similarity measures are applied between the shared context vectors of the two *language elements*.[16]

Following Lin (1998) the informativeness between the term-feature representation that measures how likely it is that t_i and t_j co-occur, is computed based on Equation 3.1 and has been introduced by Hindle (1990):

$$Lin^{'}s\ I(\langle t_i, (t_j, r_k)\rangle) = \frac{|\langle t_i, (t_j, r_k)\rangle| \cdot |\langle *, (*, r_k)\rangle|}{|\langle t_i, (*, r_k)\rangle| \cdot |\langle *, (t_j, r_k)\rangle|} \tag{3.1}$$

Applying the information from the equation above, the similarity between *language elements* can then be computed with the following formula:

[16] see Bordag (2008); Evert (2005); Weeds et al. (2004); Curran and Moens (2002); Kiela and Clark (2014) for a detailed description of different measures for mutual information and similarity measures.

$$sim(t_i, t_j) = \frac{\sum\limits_{(t_l, r_m)}^{\langle t_i,(*,*)\rangle \cap \langle t_j,(*,*)\rangle} I(\langle t_i, (t_l, r_m)\rangle) + I(\langle t_j, (t_l, r_m)\rangle)}{\sum\limits_{(t_l, r_m)}^{\langle t_i,(*,*)\rangle} I(\langle t_i, (t_l, r_m)\rangle) + \sum\limits_{(t_l, r_m)}^{\langle t_j,(*,*)\rangle} I(\langle t_j, (t_l, r_m)\rangle)}. \tag{3.2}$$

Again, we show similarities computed on the example newspaper corpus. In order to retrieving the syntactic dependencies, we apply the Stanford Parser (de Marneffe et al., 2006)[17]. As most dependency parsers also identify the POS tags, we use them to lemmatize words. Using POS alleviates that words of different POS (e.g. the word *drive* could be a noun as well as a verb) are mixed within its similarities. Results for the Lin's similarity are presented in Table 3.21. According to these similarities, we observe similarities that are closer to near-synonymy for the words searched. But we also observe counterparts and related terms, e.g. for *pitcher*, the person throwing a ball, the counterpart *hitter* and the player hitting the ball. This confirms the intuition that using language-dependent information, here syntactic dependencies and POS tags, pitches the similar words closer to near-synonymy similarities rather than relatedness.

pitcher#NN	1.00	basket#NN	1.00	NHL#NP	1.00	beer#NN	1.00	Bush#NP	1.00
hitter#NN	0.09	birdie#NN	0.05	NBA#NP	0.10	wine#NN	0.08	Gore#NP	0.15
catcher#NN	0.08	bag#NN	0.05	NFL#NP	0.09	drink#NN	0.05	McCain#NP	0.13
player#NN	0.06	touchdown#NN	0.04	NCAA#NP	0.08	alcohol#NN	0.05	Clinton#NP	0.11
baseman#NN	0.06	salad#NN	0.04	Baseball#NP	0.08	food#NN	0.05	Bradley#NP	0.10
reliever#NN	0.06	possession#NN	0.04	league#NN	0.07	coffee#NN	0.04	governor#NN	0.08
quarterback#NN	0.06	shot#NN	0.03	Football#NP	0.06	milk#NN	0.04	Dole#NP	0.08
starter#NN	0.06	net#NN	0.03	Major#NP	0.06	furniture#NN	0.04	he#PRP	0.08
guy#NN	0.06	block#NN	0.03	League#NP	0.05	cigarette#NN	0.04	Giuliani#NP	0.07
rookie#NN	0.05	bottle#NN	0.03	SEC#NP	0.05	Coke#NP	0.04	candidate#NN	0.07

Table 3.21: Most similar terms based on Lin's similarity measure.

3.2.4 Word Embeddings

Lastly we introduce *word embeddings*, which can be reduced to LSA (Levy and Goldberg, 2014a) and represents words using a dense numeric vector space. In contrast to LSA, *word embeddings* are computed based on neural networks. Whereas the idea of using neural networks for word representation is not a new idea, it previously has not been scalable to large data. Mikolov et al. (2013) present a method for achieving vector space representations, which are scalable to large amounts of data.[18]

From a general perspective, the neural network has an input layer x, zero, one or several hidden layers h and an output layer y. Using hidden layers, a neural network is capable to approximate any computable function. In the first step, each term is considered as a random

[17] `http://nlp.stanford.edu/software/lex-parser.shtml`

[18] The implementation called `word2vec` is available at `http://code.google.com/p/word2vec`.

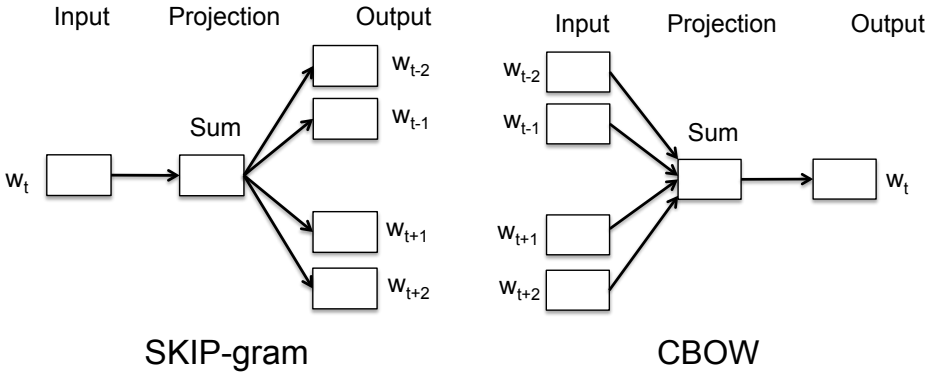


Figure 3.5: Two different embedding representation introduced by Mikolov et al. (2013). On the left-hand side the SKIP-gram representation is shown and on the right-hand side the CBOW model.

vector of length N, which describes the weights of the term to the hidden layer. Two different models are available which rely on different context presentations: the SKIP-gram and CBOW model. In both approaches, the word representations are trained unsupervised, but the input and output differ in both approaches.

In the SKIP-gram model for each word (w_t in left graph in Figure 3.5) a representation is learned that is capable to predict the previous and following words $w_{t-2}, w_{t-1}, w_{t+1}, w_{t+2}$. The CBOW model works other way around. The projection vector is learned by predicting a word using the previous and following words (see right graph in Figure 3.5). According to Mikolov et al. (2013) CBOW can be computed much faster than the SKIP-gram model but the similarities of the SKIP-gram model achieve better results in tasks for the detection of word relations.

Again, we compute similarities for the example terms using both the CBOW (see Table 3.22) and SKIP-gram (see Table 3.23) representation. We observe that the similarities for the

pitcher	1.00	basket	1.00	NHL	1.0	beer	1.00	Bush	1.00
catcher	0.66	jumper	0.61	league	0.60	sipped	0.53	Gore	0.64
reliever	0.64	baseline	0.59	NBA	0.58	soda	0.52	McCain	0.64
right-hander	0.64	layup	0.58	MLS	0.56	bottles	0.51	governor	0.58
left-hander	0.63	3-pointer	0.57	Jaromir	0.55	drink	0.51	Quayle	0.55
pitchers	0.63	perimeter	0.56	NFL	0.54	candy	0.50	Jeb	0.54
shortstop	0.59	baskets	0.55	BC-BKN-SUNS	0.53	gin	0.49	Voinovich	0.54
starter	0.59	alley-oop	0.55	Giguere	0.51	whiskey	0.49	Rove	0.52
left-handed	0.58	jumpers	0.55	Hockey	0.50	liquors	0.48	Bradley	0.50
hitter	0.58	dunk	0.53	Gauthier	0.50	Oktoberfest	0.48	Mauro	0.50

Table 3.22: Similar terms based on five example terms using the CBOW word embedding representation.

SKIP-gram and the CBOW model are very similar but the words appear at different ranks. The similar words mostly stay in the domain of the given term. In addition, for the similar terms to *beer* we mostly observe alcohol-related terms. Whereas for *basket* we observe only basketball

related terms and for pitcher only baseball related terms, for the term *NFL* we observe also terms from different sports e.g. football (*NFL*) or soccer (*MLS*).

pitcher	1.00	basket	1.00	NHL	1.0	beer	1.00	Bush	1.00
catcher	0.65	jumper	0.62	league	0.61	bottles	0.51	McCain	0.64
right-hander	0.65	layup	0.57	NBA	0.59	candy	0.51	Gore	0.62
pitchers	0.64	baskets	0.56	regular-season	0.51	soda	0.51	governor	0.57
reliever	0.64	perimeter	0.56	NFL	0.51	sipped	0.50	Rove	0.56
left-hander	0.63	3-pointer	0.55	MLS	0.51	six-pack	0.50	Voinovich	0.55
shortstop	0.61	alley-oop	0.55	Jaromir	0.50	whiskey	0.49	Jeb	0.55
right-handed	0.59	baseline	0.55	Kings	0.50	drink	0.49	Quayle	0.52
bullpen	0.59	dunk	0.53	Flyers	0.50	pub	0.49	Cheney	0.51

Table 3.23: Term similarities for five selected terms based on a SKIP-gram word embedding representation.

Here, we also observe words with different POS. Comparing the similarities to the ones from Lin's DT, only marginal overlap of the similar terms is observed. Levy and Goldberg (2014b) generalized the approach to cope with each kind of context representation. They show improvements when also adding syntactic information by using dependency parses.

3.3 Conclusion

In summary, we have introduced different models suited to compute similarities/relatedness between terms and documents. We have exemplified the results and problems relying on 10,000 newspaper articles and presented results based on 10 documents, two for different domains as well as a domain related term for inspecting term similarity.

We observe that based on a rather small corpus we achieve the best match between topically similar documents using a document-based VSM when weighting terms with tf-idf. The same result is achieved using LSA with 1000 dimensions but only when normalizing the terms by the document length. Even better results are achieved with LDA, where the same results can be achieved without using any normalization and with only 100 dimensions.

Summarizing the word similarity computation, we observe that without filtering frequent and rare terms, the document-based VSM favors similarities to rare terms. When filtering we observe topic related terms as most similar as well as morphological variations of the searched term. Whereas for LSA the similarities between the five selected terms are topical relatedness, we achieve similarities closer to near-synonymy when using term-based VSM. As expected with LDA, we extract topical similarities. For the terms in the political and sport domain, we observe rare terms, which are often names. The most similar words for *beer* are related to drinking and eating. Whereas using first order co-occurrence for extracting co-occurring term, we can observe also a topical relatedness, especially for terms, which often co-occur in enumerations, like the terms *Bush* and *beer*. Computing similarities between terms based on the

neighboring terms, we observe again topical related terms. In contrast to the document-based approach, more senses of a term are introduced into the similar terms. Additionally, many terms with the closest similarity distance are not always related to the terms, we computed the similarities. This is alleviated using Lin's algorithm for computing similarities between terms. His algorithm uses dependency parses and their dependency relation name as context representation and applies a weighting between terms and contexts. The two different methods used for computing word embeddings are not too different, which might be rooted by the rather small corpus. Again, the similarities are topically related and are most similar to the ones extracted with LDA. From the findings from this section we obtain the similarities that are closest to near-synonymy using Lin's approach, which uses syntactic dependency parses for the similarity computation and does not rely on the cosine similarity.

In this section we have introduced both symbolic and non-symbolic approaches for computing similarities. In symbolic approaches a term is represented by "context symbols" (e.g. words, syntactic dependencies). This enables the reasoning for similarities, as the responsible context for the similarity of two terms is still interpretable. Furthermore, the context representation of symbolic approaches can easily be extended further contexts in order to compute similarities (see Chapter 6). In this chapter the VSM models, both term- and document-based, Lin's DT and the co-occurrences can be classified as symbolic approaches, as the context can still be extracted. However, storing the terms and context features in a matrix, results in a sparse matrix. In order to alleviate some of the sparsity issues additional pruning of infrequent terms and weighting methods are required. Operating on sparse matrices becomes time and memory consuming, as the matrix gets huge due to many words that occur only rarely (see Zipf's law in Section 2.6). A compact matrix representation is achieved with non-symbolic approaches, which represent the context of a word in a dense numeric vector representation. Here, we have described LSA, LDA and word embeddings, which we assign to non-symbolic methods. Whereas these approaches provide good results, their vector representations are not interpretable and make it hard to be used straightforwardly. In chapter 5, we propose a new symbolic method for computing term similarities, which uses a compact representation and does not face sparsity issues. We conclude this chapter with a summary of the findings for each model, presented in Table 3.24.

	similarities		type of			
	terms	document	similarity	runtime	sparsity	symbolic
doc.-based VSM	yes	yes	relatedness	medium	high	yes
LSA	yes	yes	relatedness	high	low	no
LDA	yes	yes	relatedness	medium	low	no
co-occurrences	yes	no	no similarity/relatedness	low	low	yes
term-based VSM	yes	no	relatedness/near-synonymy	medium	high	yes
Lin's DT	yes	no	near-synonymy	high	low	yes
SKIP-gram	yes	no	topical/near-synonymy	medium	no	no
CBOW	yes	no	topical/near-synonymy	low	no	no

Table 3.24: This table summarized properties of the semantic models we described in this chapter.

Chapter 4

Text Segmentation Using Topic Models

> Everything should be made as simple as possible, but no simpler.
>
> - Albert Einstein

Segmentation of natural language is a mandatory preprocessing step. Whereas the segmentation of tokens can be considered as solved on token level for Western languages, the segmentation into topical coherent chunks is more complicated. The segmentation of topically coherent chunks is a major step, which humans perform for understanding language and words. This information can be used to enrich the contextual representation for terms. In this chapter, we use the statistical topic model LDA (see Section 3.1.3) for segmenting raw text into topically coherent segments.

This chapter is organized as follows: The first section introduces the task of Text Segmentation (TS) and is followed by an overview of several TS algorithms. Then we introduce a method where we replace words with topic IDs obtained by a topic model. We lay out three algorithms using these topic IDs in detail and show improvements for the topic-based variants. Section 4.5 evaluates parameters for the topic model in combination with parameters of our TopicTiling algorithm. This provides information about the different parameters of the topic model used. In Section 4.6 we demonstrate the performance of our algorithm using two datasets. This chapter is based on the following publications: Riedl and Biemann (2012a,b,c)

4.1 Introduction

TS is concerned with "automatically break[ing] down documents into smaller semantically coherent chunks" (Jurafsky and Martin, 2009). We expect that semantically coherent chunks are also similar in a topical sense. Thus, we view a document as a sequence of topics. This

semantic information can be modeled using TMs. TS is realized by an algorithm that identifies topical changes in the sequence of topics.

TS is an important task, needed for various Natural Language Processing (NLP) tasks, e.g. information retrieval and text summarization. In information retrieval tasks, TS can be used to extract segments of the document that are topically interesting (Llopis et al., 2002). In text summarization, segmentation results are important to ensure that the summarization covers all themes a document contains (Angheluta et al., 2002). Another application could be a writing aid to assist authors with possible positions for subsections.

In this chapter, we use the Latent Dirichlet Allocation (LDA) topic model (Blei et al., 2003). We show that topic IDs, assigned to each word in the last iteration of the Bayesian inference method of LDA, can be used to improve TS significantly in comparison to methods using word-based features. This is demonstrated on three algorithms: TextTiling (Hearst, 1997), C99 (Choi, 2000) and the introduced algorithm called TopicTiling. TopicTiling resembles TextTiling, but is conceptually simpler since it does not have to account for the sparsity of word-based features.

In a sweep over parameters of LDA and TopicTiling, we find that using topic IDs of a single last inference iteration leads to enormous instabilities with respect to TS error rates. These instabilities can be alleviated by two modifications: (i) repeating the inference iterations several times and selecting the most frequently assigned topic ID for each word across several inference runs, (ii) storing the topic IDs assigned to each word for each iteration during the Bayesian inference and selecting most frequently assigned topic ID (the mode) per word. Both modifications lead to similar stabilization, however (ii) needs less computational resources. Furthermore, we can also show that the standard parameters recommended by Griffiths and Steyvers (2004) do not always yield optimal results.

Using what we have learned in these series of experiments, we evaluate the performance of an optimized version of TopicTiling on two datasets: The Choi dataset (Choi, 2000) and a more challenging WSJ corpus provided by Galley et al. (2003). On both dataset we observe state-of-the-art segmentation results. In comparison to most recent TS algorithms, TopicTiling performs the segmentation in linear time.

4.2 Related Work

Topic segmentation can be divided into two sub-fields: (i) linear topic segmentation and (ii) hierarchical topic segmentation. Whereas linear topic segmentation deals with the sequential analysis of topical changes, hierarchical segmentation concerns with finding more fine-grained subtopic structures in texts.

Hearst (1997) introduced one of the first unsupervised linear topic segmentation algorithms: TextTiling segments texts in linear time by calculating the similarity between two blocks of words based on the cosine similarity. The calculation is accomplished by two vectors containing the number of occurring terms of each block. LcSeg, a TextTiling-based algorithm,

was published by Galley et al. (2003). In comparison to TextTiling, it uses *tf-idf* for weighting terms, which improves TS results. The algorithm C99 was presented by Choi (2000) and uses a matrix-based ranking and a clustering approach in order to relate the most similar textual units. Similar to the previous introduced algorithms, C99 uses words.

Utiyama and Isahara (2001) presented one of the first probabilistic approaches using Dynamic Programming (DP) that is called U00. DP is a paradigm that can be used to efficiently find paths of minimum cost in a graph. Text Segmentation algorithms using DP, represent each possible segment (e.g. every sentence boundary) as an edge. Providing a cost function that penalizes common vocabulary across segment boundaries, DP can be applied to find the segments with minimal cost.

Related to our work are a modified C99 algorithm, introduced by Choi et al. (2001) that uses the term-representation matrix in latent space of LSA in combination with a term frequency matrix to calculate the similarity between sentences and two DP approaches described by Misra et al. (2009) and Sun et al. (2008): here, topic modeling is used to alleviate the sparsity of word vectors. The algorithm of Sun et al. (2008) follows the approach described by Fragkou et al. (2004), but uses a combination of topic distributions and term frequencies. A Fisher kernel is used to measure the similarity between two blocks, where each block is represented as a sequence of sentences. The kernel uses a measure that indicates how many topics two blocks share, combined with the term frequency, which is weighted by a factor that determines how likely the terms belong to the same topic. They use entire documents as blocks and generate the topic model using the test data. This method is evaluated using an artificially garbled Chinese corpus. Similarly to our approach, Misra et al. (2009) extended the DP algorithm U00 from Utiyama and Isahara (2001) with information from topic models. Instead of using the probability of word co-occurrences, they use the probability of co-occurring topics. Segments with many topics have a low topic-probability, which is used as a cost function in their DP approach. Misra et al. (2009) trained the topic model on a collection of the Reuters corpus and a subset of the Choi dataset, and tested on the remaining Choi dataset. The topics for this test set have to be generated for each possible segment using Bayesian inference methods, resulting in high computational cost. In contrast to these previous DP approaches, we present a computationally more efficient solution. Du et al. (2010) presents another text segmentation algorithm, which extends the LDA topic model. Further extensions of their methods have been proposed by Du et al. (2013).

Another approach for text segmentation is the usage of Hidden Markov Model (HMM) (van Mulbregt et al., 1998). Blei and Moreno (2001) apply an Aspect Hidden Markov Model (AHMM), which combines an aspect model (Hofmann, 1999) with a HMM. The limiting factor of both approaches is that a segment is assumed to have only one topic. Gruber et al. (2007) solved this issue by extending LDA to consider the word and topic ordering using a Markov Chain. In contrast to LDA, not a word is assigned to a topic, but a sentence, so the segmentation can be performed sentence-wise.

In early TS evaluations, Hearst (1994) measured the fitting of the estimated segments using precision and recall. However, these measures are not appropriate for the task, since the distance of a falsely estimated boundary to the correct one is not considered at all. With P_k (Beeferman et al., 1999), a measure was presented that regulates this problem. But there are issues concerning the P_k measure, as it uses an unbalanced penalizing between false negatives and false positives. WindowDiff (WD) (Pevzner and Hearst, 2002) solves this problem, but most published algorithms still use the P_k measure. In practice, both measures are highly correlated. While there are newer published metrics (see Georgescul et al. (2006), Lamprier et al. (2007), Scaiano and Inkpen (2012) and Kazantseva and Szpakowicz (2012)), in practice still the two metrics P_k and WD are commonly used.

To handle near misses, P_k uses a sliding window with a length of k tokens, which is moved over the text to calculate the segmentation penalties. This leads to following pairs: $(1, k), (2, k+1), ..., (n-k, n)$, with n denoting the length of the document. For each pair (i, j) it is checked whether positions i and j belong to the same segment or to different segments. This is done separately for the gold standard boundaries and the estimated segment boundaries. If the gold standard and the estimated segments do not match, a penalty of 1 is added. Finally, the error rate is computed by normalizing the penalty by the number of pairs $(n-k)$, leading to a value between 0 and 1. A value of 0 denotes a perfect match between the gold standard and the estimated segments. The value of parameter k is assigned to half of the number of tokens in the document divided by the number of segments, given by the gold standard.

According to Pevzner and Hearst (2002), a drawback of the P_k measure is its unawareness of the number of segments between the pair (i, j). WD is an enhancement of P_k: the number of segments between position i and j are counted, where again the distance between the positions is specified by parameter k. Then the number of segments is compared between the gold standard and the estimated segments. If the number of segments is not equal, 1 is added to the penalty, which is then normalized by the number of pairs to receive an error rate between 0 and 1.

Yaari (1997) proposed the first hierarchical algorithm that uses the cosine similarity and a agglomerative clustering approaches. A hierarchical Bayesian algorithm based on LDA was presented by Eisenstein (2009). In our work, however, we focus on linear topic segmentation.

The generative model called LDA was introduced by Blei et al. (2003) and discovers topics based on a training corpus. Model training estimates two distributions: A topic-word distribution and a topic-document distribution. As LDA is a generative probabilistic model, the creation process follows a generative story: First, for each document a topic distribution is sampled. Then, for each document, words are randomly chosen, following the previously sampled topic distribution. Using the Gibbs inference method, LDA is used to apply a trained model for unseen documents. Here, words are annotated by topic IDs by assigning the most probable topic ID based on the two distributions. Note that the inference procedure, in partic-

ular, marks the difference between LDA and earlier dimensionality reduction techniques such as Latent Semantic Analysis.

4.3 Text Segmentation Datasets

In this work, we use two datasets: A document collection generated based on the Brown corpus (Francis and Kučera, 1964) and a more challenging corpus generated from articles of the Wall Street Journal (WSJ).

4.3.1 Choi Dataset

The Choi dataset (Choi, 2000) is commonly used in the field of TS (see e.g. Misra et al. (2009); Sun et al. (2008); Galley et al. (2003)). It is an artificially generated corpus generated from the Brown corpus and consists of 700 documents. Each document consists of ten segments. The document generation was performed extracting consecutive snippets of 3-11 sentences from different documents from the Brown corpus. There are 400 documents with segment lengths of 3-11 sentences and 100 documents, each with sentence counts of 3-5, 6-8 and 9-11.

4.3.2 Galley Dataset

Galley et al. (2003) present two corpora for written language, each corpus contains 500 documents, which are also generated artificially. In comparison to Choi's dataset, the segments in the 'documents' vary from 4 to 22 segments and are generated by concatenating full source documents. Using full documents makes this corpus a more realistic one in comparison to the one provided by Choi. One dataset is generated from the WSJ documents of the Penn Treebank (PTB) project (Marcus et al., 1994) and the other dataset is based on Topic Detection Track (TDT) documents (Wayne, 1998). As the WSJ dataset seems to be harder for the task of TS than the Choi dataset (consistently higher error rates across several works), we use this dataset for experimentation.

4.4 From Words to Topics

4.4.1 Method to Represent Words with Topic IDs

The method (see also Misra et al., 2009; Sun et al., 2008) for using information achieved by topic models is conceptually simple: Instead of using words directly as features to characterize textual units, we use their topic IDs as assigned by Bayesian inference. LDA inference assigns a topic ID to each word in the test document in each inference iteration step, based on a TM

trained on a training corpus. The first series of experiments use the topic IDs assigned to each word in the last inference iteration. Figure 4.1 depicts the general setup.

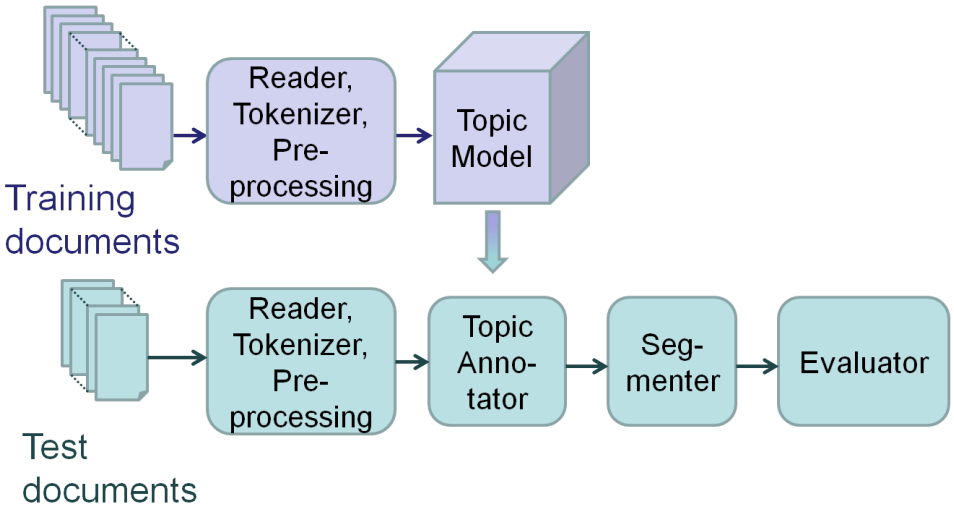


Figure 4.1: Basic concept of text segmentation using Topic Model.

First, preprocessing steps[19] like tokenizing, sentence segmentation, POS tagging[20] or filtering are applied to the training and test documents.

The training data used to estimate the topic models should ideally be from the same domain as the test documents. Even though topic models belong to the unsupervised learning paradigm, no information about the test data should inform the training and thus, no test documents should be used for the topic model estimation. The topic model is estimated once in advance and can then be used for inference on the test documents. In this step, the LDA inference assigns a topic ID to each word in the test document and generates a document topic distribution.

An example of a text annotated with topic IDs, taken from the WSJ test data, is presented in Figure 4.2. One can clearly see the boundary by looking at the most probable topic IDs. The first text is about a telecommunication company, having mostly topic ID 2 assigned to words. The second segment is about an anti-government rally in South Africa. Most words of this segment are annotated with topic ID 37. The topic IDs are not assigned statically per word, but converge from Gibbs Sampling inference, which iterates over the words and re-samples topic IDs according to the per-document topic distribution and the per-topic word probabilities from the previous inference step. For example, the word *people* (marked bold in Figure 4.2) is marked with topic 37 since this topic is highly probable in the document. Using this word in a different context would most likely lead to a different topic ID.

[19]We use the DKPro framework (de Castilho and Gurevych, 2014) available at `https://github.com/dkpro/dkpro-core`.

[20]For POS tagging we use the TreeTagger (Schmid, 1994, 1995).

Mr:62 .:97 Pohs:2 ,:2 previously:4 executive:2 vice:2 president:2 and:17 chief:2 operating:2 officer:2 ,:72 was:2 named:2 interim:2 president:2 and:73 chief:2 executive:2 officer:2 after:17 David:2 M:27 .:36 Harrold:65 ,:2 a:84 company:2 founder:2 ,:26 resigned:2 from:91 the:34 posts:2 for:62 personal:61 reasons:2 in:84 August:2 .:58 Cellular:70 said:54 Robert:2 J:61 .:42 Lunday:2 Jr:18 .:31 ,:44 its:57 chairman:2 and:73 another:25 founder:2 ,:31 resigned:2 from:91 the:57 company:2 's:24 board:2 to:10 pursue:2 the:10 sale:55 of:67 his:28 telephone:31 company:42 ,:74 Big:10 Sandy:50 Telecommunications:31 Inc:2 .:74
APARTHEID:37 FOES:37 STAGED:41 a:37 massive:37 anti-government:37 rally:37 in:40 South:37 Africa:37 .:19 More:29 than:34 70:45 ,:26 000 **people:37** *filled:17 a:22 soccer:37 stadium:88 on:46 the:34 outskirts:37 of:93 the:24 black:37 township:37 of:45 Soweto:37 and:37 welcomed:11 freed:37 leaders:37 of:98 the:57 outlawed:37 African:37 National:45 Congress:87 .:72 It:79 was:55 considered:37 South:37 Africa:37 's:33 largest:90 opposition:67 rally:37 .:37*

Figure 4.2: Excerpt from a test document for text segmentation, taken from Galley's WSJ corpus. Each word is followed by a colon and a number, which represents the topic ID.

In the example, all tokens are used for topic model estimation — it is also possible to filter tokens by part of speech or very short sentences for the purpose of model estimation and inference. This is expected to lead to even sparser topic distributions.

Once the topic IDs are assigned, most previous segmentation algorithm can be applied, using the topic ID of each word instead of the word itself.

In this work, we implement topic-based versions of C99 (Choi, 2000), TextTiling (Hearst, 1994) and develop a TextTiling-based method called TopicTiling. Our aim is to find a simplified algorithm that solves the text segmentation problem, using topic IDs.

4.4.2 Text Segmentation Algorithms Using Topic Models

C99 Using Topic Models

The topic-based version of the C99 algorithm (Choi, 2000), called C99LDA, divides the input text into minimal units on sentence boundaries. A similarity matrix $S_{m \times m}$ is computed, where m denotes the number of units (sentences). Every element s_{ij} is calculated using the cosine similarity (e.g. Manning and Schütze, 1999, p. 299) between unit i and j. For these calculations, each unit i is represented as a T-dimensional vector, where T denotes the number of topics selected for the topic model. Each element t_k of this vector contains the number of times topic ID k occurs in unit i. Next, a rank matrix R is computed to improve the contrast of S: Each element r_{ij} contains the number of neighbors of s_{ij} that have lower similarity scores than s_{ij} itself. This step increases the contrast between regions in comparison to matrix S. In a final step, a top-down hierarchical clustering algorithm is performed to split the document into m segments. This algorithm starts with the whole document considered as one segment and splits off segments until the stop criteria are met, e.g. the number of segments or a similarity

threshold. At this, the ranking matrix is split at indices i, j that maximize the inside density function D.

$$D = \sum_{k=1}^{m} \frac{\text{sum of ranks within segment } k}{\text{area within segment } k} \tag{4.1}$$

As a threshold-based criterion, the gradient δD is introduced as $\delta D^{(n)} = D^{(n)} - D^{(n-1)}$. The threshold can then be calculated by $\mu + c \times \sigma$, where mean μ and the standard deviation σ are calculated from the gradients and c is a constant, which we set to 1.2 as regulated by Choi (2000).

TextTiling Using Topic Models

In TTLDA, the topic-based version of TextTiling (TT) (Hearst, 1994), documents are represented as a sequence of n topic IDs instead of words. TTLDA splits the document into *topic-sequences*, instead of sentences, where each sequence consists of w topic IDs. To calculate the similarity between two topic-sequences, called *sequence-gap*, TTLDA uses k topic-sequences, named *block*, to the left and to the right of the sequence gap. This parameter k defines the so-called *blocksize*. The cosine similarity is applied to compute a similarity score based on the topic frequency vectors of the adjacent blocks at each sequence-gap. A value close to 1 indicates a high similarity between two blocks and a value close to zero denotes a low similarity. The sharpness for each sequence-gap is measured by the *depth score* d_i that is given by:

$$d_i = 1/2(hl(i) - s_i + hr(i) - s_i). \tag{4.2}$$

The function $hl(i)$ returns the highest similarity score on the left side of the sequence-gap index i that does not increase and $hr(i)$ returns the highest score on the right side. Then all local maxima positions are searched based on the depth scores.

In the next step, these obtained maxima scores are sorted. If the number of segments n is given as input parameter, the n highest depth scores are used. Otherwise, a cut-off function is used that applies a segment only if the depth score is larger than $\mu - \sigma/2$. The mean μ and the standard deviation σ are calculated based on the entirety of depth scores. As TTLDA calculates the depth on every topic-sequence using the highest gap, this could result to a segment in the middle of a sentence. In order to avoid such an erroneous segmentation, a final step ensures that the segmentation is positioned at the nearest sentence boundary.

TopicTiling

The two previously described algorithms are modifications of existing algorithms, where we replaced words with topic IDs. Here we introduce a Text Segmentation algorithm called Top-

icTiling[21]. This algorithm is based on TextTiling, but is conceptually simpler. TopicTiling assumes a sentence s_i as the smallest basic unit. Between each position p between two adjacent sentences, a *coherence score* c_p is calculated. To calculate the coherence score, we exclusively use the topic IDs assigned to the words by inference: Assuming an LDA model with T topics, each block is represented as a T-dimensional vector. The t-th element of each vector contains the frequency of the topic ID t obtained from the respective block. The coherence score is calculated by cosine similarity for each adjacent "topic vector". Values close to zero indicate

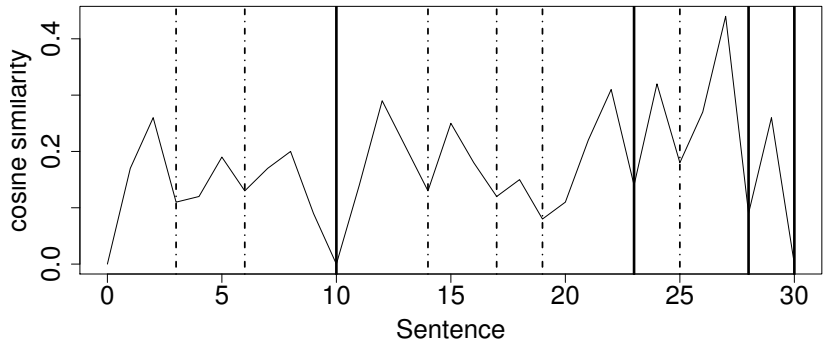


Figure 4.3: Similarity scores plotted for a document used for text segmentation. The vertical dotted lines indicate all possible segment boundaries. The solid lines indicate segments chosen by the threshold criterion when the number of segments is not given in advance.

marginal relatedness between two adjacent blocks, whereas values close to one denote a substantial connectivity. Next, the coherence scores are plotted to trace the local minima (see Figure 4.3). These minima are utilized as possible segmentation boundaries. Instead of using the c_p values itself, a *depth score* d_p is calculated for each minimum (cf. TextTiling, Hearst (1997)). In comparison to TopicTiling, TextTiling calculates the depth score for each position and then searches for maxima. The depth score measures the deepness of a minimum by looking at the highest coherence scores on the left and on the right and is calculated using Equation 4.3 (cf. Equation 4.2 for the depth score in the previous section).

$$d_p = 1/2(hl(p) - c_p + hr(p) - c_p) \tag{4.3}$$

We illustrate the workings of the function hl (the highest peak on the left side) and hr (the highest peak on the right side) in Figure 4.4. The function $hl(p)$ iterates to the left as long as the score increases and returns the highest coherence score value. With the $hr(p)$ function we search for the highest coherent score on the right side. According to the illustration, $hl(4) = 0.93$, the score value at position 2, and $hr(4) = 0.99$ from the value at position 7.

[21] An implementation of TopicTiling is available for download at: `http://sourceforge.net/projects/topictiling/`.

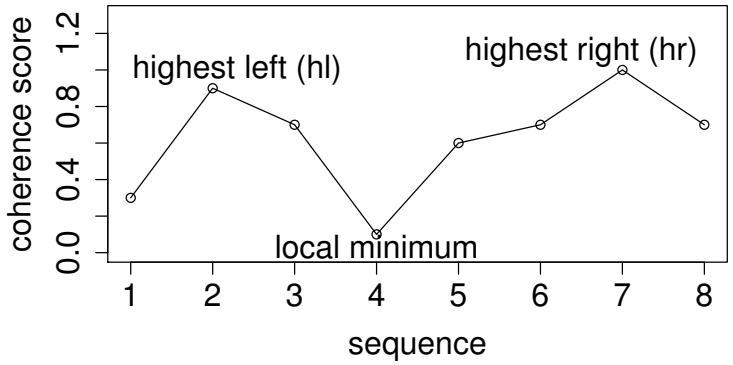


Figure 4.4: Illustration of the highest left (hl) and the highest right (hr) peak according to a local minimum. These peaks are then used to compute the depth score.

If the number of segments n is given as input, the n highest depth scores are used as segment boundaries. Otherwise, a threshold is applied (cf. TextTiling). This threshold predicts a segment if the depth score is larger than $\mu - \sigma/2$, with μ being the mean and σ being the standard variation calculated on the depth scores.

The algorithm runtime is linear in the number of possible segmentation points, i.e. the number of sentences. For each segmentation point, the two adjacent blocks are sampled separately and combined into the coherence score. This is not achieved with dynamic programming approaches for TS as described by Utiyama and Isahara (2001); Misra et al. (2009).

4.4.3 Experiment: Word-based vs. Topic-based Methods

To show the advantage of the topic-based representation introduced in Section 4.4.1, we present results for TT and C99 using words and topic IDs, and for TopicTiling.

Experimental Setup

As laid out in Section 4.4.1, an LDA Model is estimated on a training dataset and used for inference on the test set. We perform a 10-fold cross-validation (CV) for all reported results, to ensure that we do no use information from the test set. In order to reduce the variance, stemming from the random nature of sampling and inference, the results for each fold are calculated 30 times using different LDA models.

While we aim at not using the same *documents* for training and testing by using a folded CV scheme, it is not guaranteed that all testing data is unseen, since the same source *sentences* can find their way in several artificially crafted *documents*. We have detected that all *sentences* from the training subset also occur in the test subset, but not in the same combinations. This makes the Choi dataset artificially easy for supervised approaches. However, this problem affects all

evaluations on this dataset that use any kind of training, be it LDA models (Misra et al., 2009) or *tf-idf* values (Fragkou et al., 2004; Galley et al., 2003).

The LDA model is trained with $T = 100$ topics, 500 sampling iterations and symmetric hyperparameters as recommended by Griffiths and Steyvers (2004)[22]. Unseen data is annotated with topic information, using LDA inference, sampling $i = 100$ iterations. Inference is executed sentence-wise, since sentences form the minimal unit of our segmentation algorithms and we cannot use document information in the test setting. The performance of the algorithms is measured using P_k and WindowDiff (*WD*) metrics, cf. Section 4.2 The C99 algorithm is initialized with an 11×11 ranking mask, as recommended by Choi (2000). TT is configured according to Choi (2000) with sequence length $w = 20$ and block size $k = 6$.

Results

The experiments are executed in two settings based on the C99 and TT implementations[23]: using words (C99, TT) and using topics (C99LDA, TTLDA). TT and C99 use stemmed words and filter out words using a stopword list. C99 additional removes words using predefined regular expressions. In the case of topic-based variants, no stopword filtering or stemming was deemed necessary. Table 4.1 presents the result of the different algorithms. The results in column two and three are achieved when the number of segments is known beforehand and column four and five show results when the number of segments is estimated automatically.

Method	**known number of segments**		**unknown number of segments**	
	P_k	*WD*	P_k	*WD*
C99	11.20	12.07	12.73	14.57
C99LDA	4.16	4.89	8.69	10.52
TT	44.48	47.11	49.51	66.16
TTLDA	1.85	2.10	16.41	21.40
TopicTiling	2.65	3.02	4.12	5.75
TopicTiling (filtered)	**1.50**	**1.72**	**3.24**	**4.58**

Table 4.1: This table shows results based on Choi's text segmentation dataset. We show P_k values for different segment length for TT with words and topics (TTLDA), C99 with words and topics (C99LDA) and TopicTiling using all sentences and using only sentences with more than 5 word tokens (filtered).

We note that *WD* values are always higher than the respective P_k values. However, we also observe that these measures are highly correlated. First, we discuss results for the setting with number of segments provided (see columns 2-3 of Table 4.1). A significant improvement for C99 and TT can be achieved when using topic IDs. In case of C99LDA, the error rate is at

[22]We set the hyperparameter as follows: $\alpha = 50/N$ and $\beta = 0.01$. For computing the LDA model, we use the JGibbLda implementation, which is available at `http://jgibblda.sourceforge.net/`.

[23]We use the implementations by Choi available at `http://code.google.com/p/uima-text-segmenter/`.

least halved and for TTLDA the error rate is reduced by a factor of 20. The newly introduced algorithm TopicTiling as described above does not improve over TTLDA. Analysis revealed that the Choi corpus includes also captions and other "non-sentences" that are marked as sentences, which causes TopicTiling to add false positive segments since the topic vectors are too sparse for these short "non-sentences". Thus, we filter out "sentences" with less than 5 words (see bottom line in Table 4.1). This reduces errors values in comparison to the results achieved with TTLDA.

Without the number of segments given in advance (see columns 3-4 in Table 4.1), we again observe significantly better results, comparing topic-based methods to word-based methods. Nevertheless, the error rates of TTLDA are unexpectedly high. We discovered in data analysis that TTLDA estimates too many segments, as the topic ID distributions between adjacent sentences within a segment are often too diverse, especially in face of random fluctuations from the topic assignments. Estimating the number of segments is better achieved using TopicTiling instead of TTLDA even without any additional sentence filtering. As we aimed to find a simple algorithm that can cope with the topic-based approach, we use TopicTiling for the next series of experiments.

4.5 Sweeping the Parameter Space of LDA

This section demonstrates the influence of the parameters available for the topic model LDA. Aside from the main parameter, the number of topics or dimensions T, less attention has been spent to understand the interactions of hyperparameters, the number of sampling iterations in model estimation and interference, and the stability of topic assignments across runs using different random seeds in the LDA topic model. While progress in the field of topic modeling is mainly made by adjusting prior distributions (e.g. Sato and Nakagawa, 2010; Wallach et al., 2009), or defining more complex mixture models (Heinrich, 2011), it seems unclear whether improvements, reached on intrinsic measures like perplexity or on application-based evaluations, are due to an improved model structure or could originate from sub-optimal parameter settings or due to the randomized nature of the sampling process. This has also been confirmed by Chang et al. (2009), who demonstrated that the coherence of topics is negative correlated when comparing perplexity measures to human annotations.

These subsections address these issues by systematically sweeping the parameter space and evaluating LDA parameters with respect to text segmentation results achieved by TopicTiling.

4.5.1 Experimental Setup

Again, the Choi dataset (see Section 4.3.1) is used, applying a 10-fold CV as described in Section 4.4.3. To assess the robustness of the TM, we sweep over varying configurations of the LDA model, and plot the results using Box-and-Whiskers plots: the box indicates the quartiles and

the whiskers are maximally 1.5 times Interquartile Range (IQR) or equal to the data point that has not a distance larger than 1.5 times IQR. The following parameters are subject to our exploration:

- T: Number of topics used in the LDA model. Common values vary between 50 and 500.
- α : Hyperparameter that regulates the sparseness topic-per-document distribution. Lower values result in documents being represented by fewer topics (Heinrich, 2004). Recommended: $\alpha = 50/T$ (Griffiths and Steyvers, 2004)
- β : Reducing β increases the sparsity of topics, by assigning fewer terms to each topic, which is correlated to how related words need to be, to be assigned to a topic (Heinrich, 2004). Recommended: $\beta = \{0.1, 0.01\}$ (Griffiths and Steyvers, 2004; Misra et al., 2009)
- m: Model estimation iterations. Recommended / common settings: $m = 500 - 5000$ (Griffiths and Steyvers, 2004; Wallach et al., 2009)
- i: Inference iterations. Recommended / common settings: 100
- d: Mode of topic assignments. At each inference iteration step, a topic ID is assigned to each word within a document (represented as a sentence in our application). With this option ($d = true$, we count these topic assignments for each single word in each iteration. After all i inference iterations, the most frequent topic ID is chosen for each word in a document.
- r: Number of inference runs: We repeat the inference r times and assign the most frequently assigned topic per word at the final inference iteration for the segmentation algorithm. High r values might reduce fluctuations due to the randomized process and lead to a more stable word-to-topic assignment.
- w Window: We introduce a so-called *window parameter* that specifies the number of sentences to the left and to the right of position p that define two *blocks*: $s_{p-w}, s_{p-w+1}, \dots, s_p$ and $s_{p+1}, \dots, s_{p+w}, s_{p+w+1}$.

Most of these parameters apply for the TM. Other works stabilize topic assignments by averaging assignments probed from every 50-100th iteration. Examining this effect more closely, we look at the mechanisms of using several inference runs r to find the correct segments and the mode of topic assignments d. Further, we did not find previous work that systematically varies TM parameters in combination with measures other than perplexity.

4.5.2 Parameter Sweeping Evaluation

Number of Topics T

To provide a first impression of the data, a 10-fold CV is calculated and the segmentation results are visualized in Figure 4.5. All box plots are generated from the P_k values of 700 documents.

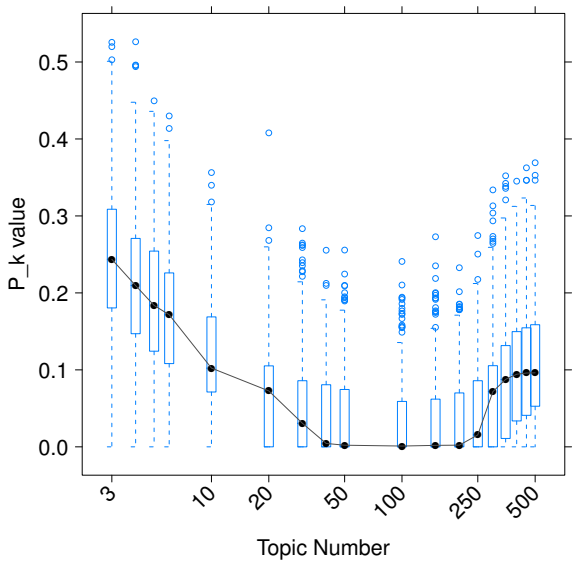


Figure 4.5: Box plots for different number of topics T. All box plots are generated from the average P_k value of 700 documents, $\alpha = 50/T$, $\beta = 0.1$, $m = 1000$, $i = 100$, $r = 1$.

As expected, there is a continuous range of topic numbers, namely between 50 and 150 topics, where we observe the lowest P_k values. Using too many topics leads to overfitting of the data and too few topics results in too general distinctions to grasp text segment information. This general picture is in line with other studies that determine an optimum for T, (cf. Griffiths and Steyvers, 2004), which is specific to the application and the dataset.

Estimation and Inference Iterations

The next step examines the robustness of the model estimation iterations m needed to achieve stable results. 600 documents are used for training an LDA model and the remaining 100 documents are segmented using this model. This evaluation is performed using 100 topics (as this number leads to stable results according to Figure 4.5) and 20 and 250 topics. To assess stability across different model estimation runs, we trained 30 LDA models using different random seeds. Each box plot in Figure 4.6 is generated from 30 mean values, calculated from the P_k values of the 100 documents. The variation indicates the score variance for the 30 different models.

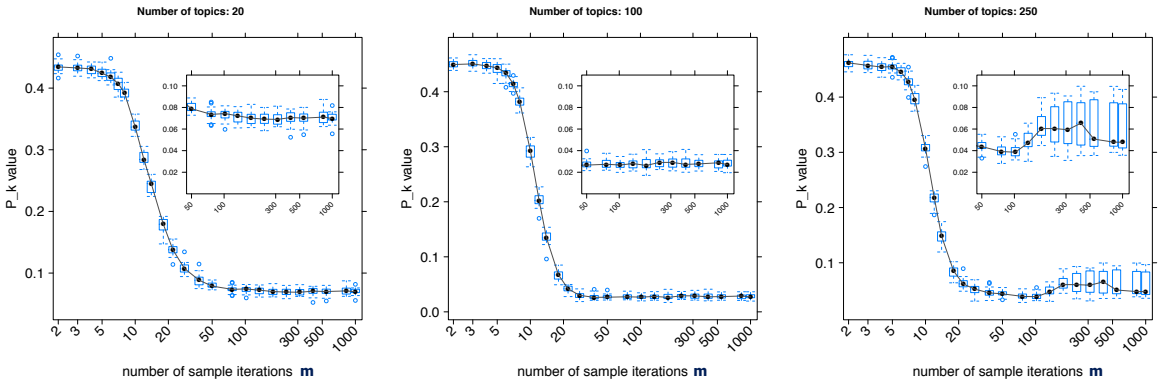
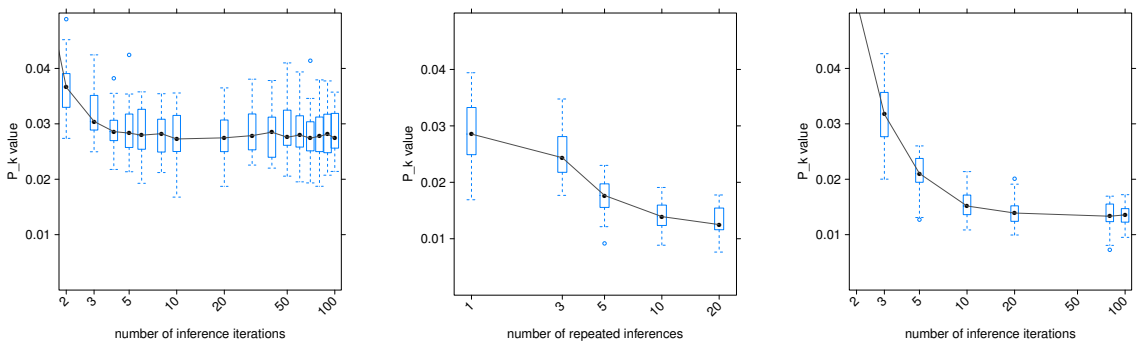


Figure 4.6: Box plots with different number of sample iterations m used to estimate the model, with T=20,100,250 (from left to right), $\alpha = 50/T$, $\beta = 0.1$, $i = 100$, $r = 1$. Each box plot is generated from 30 mean values calculated from 100 documents.

Using 100 topics (see the middle graph in Figure 4.6), the burn-in phase starts with 8–10 iterations and the mean P_k values stabilize after 40 iterations. But looking at the inset for large m values, significant variations between the different models can be observed: note that the P_k error rates are between 0.021 - 0.037. As expected, using 20 and 250 topics (see left and right graph in Figure 4.6) leads to inferior P_k scores as with 100 topics. Computing LDA with 250 topics, a robust range for the error rates can be found, as seen in the right graph in Figure 4.6, between 20 and 100 sample iterations. With more iterations m, the results become worse and unstable: as the 'natural' topics of the collection have to be split in too many topics in the model, perplexity reduction that drives the estimation process leads to random fluctuations, which the TopicTiling algorithm is sensitive to. Manual examination of models for $T = 250$ revealed that in fact many topics do not stay stable across estimation iterations. In the next step we sweep over several inference iterations i using 100 topics. Starting from 5 iterations, error rates do not change much (see Figure 4.7a). But there is still substantial variance, between about 0.019 – 0.038 for inference on sentence units.

Repeat the Inference r Times

To decrease this variance, we assign the topic not only from a singe inference run, but we repeat the inference calculations several times, denoted by the parameter r. Then the frequency of assigned topic IDs per token is counted across the r runs, and we assign the most frequent topic ID (frequency ties are broken randomly). The box plot for several evaluated values of r is shown in Figure 4.7b. This log-scaled plot shows that both variance and P_k error rate can be substantially decreased. Already for $r = 3$, we observe a significant improvement in comparison to the default setting of $r = 1$ and with increasing r values, the error rates are reduced even more: for $r = 20$, variance and error rates are cut in less than half of their original values using this simple operation.



(a) number of inferences i (b) number of repeated inference runs r (c) mode method d = true

Figure 4.7: Figure a) shows the box plots for different inference iterations i, Figure b) displays the box plots for several inference runs r and Figure c) presents the usage of the mode method $d = true$. All remaining parameters are set to the default values as described in Section 4.5.1.

Mode of Topic Assignment d

In the previous experiment, we use the topic IDs that have been assigned most frequently at the last inference iteration step. Now, we examine something similar, but for all i inference steps of a single inference run: we select the mode of topic ID assignments for each word across all inference steps. The impact of this method on error and variance is illustrated in Figure 4.7c. Using a single inference iteration, the topic IDs are almost assigned randomly. After 20 inference iterations, P_k values below 0.02 are achieved. With further iterations, the decrease of the error rate is only marginal. In comparison to the repeated inference method, the additional computational costs of this method are much lower as the inference iterations have to be carried out anyway in the default application setting. Note that this is different from using the overall topic distribution as determined by the inference step, since this winner-takes-it-all approach reduces noise from random fluctuations. As this parameter stabilizes the topic IDs at low computational costs, we recommend using this option in all setups where subsequent steps rely on single topic assignments.

Hyperparameters α and β

In many previous works (cf. Griffiths and Steyvers (2004)), hyperparameter settings $\alpha = 50/T$ and $\beta = \{0.1, 0.01\}$ are commonly used. In the next series of experiments, we investigate how different parameters of these both parameters can change the TS task. Analyzing the α values, shown in Figure 4.8, we can see that the recommended value for $T = 100$, which is $\alpha = 0.5$, results into sub-optimal results, and an error rate reduction of about 40% can be achieved by setting $\alpha = 0.1$. We observe that P_k rates and their variance are relatively stable for values of β between the recommended settings of 0.1 and 0.01. Values larger than 0.1 lead to much

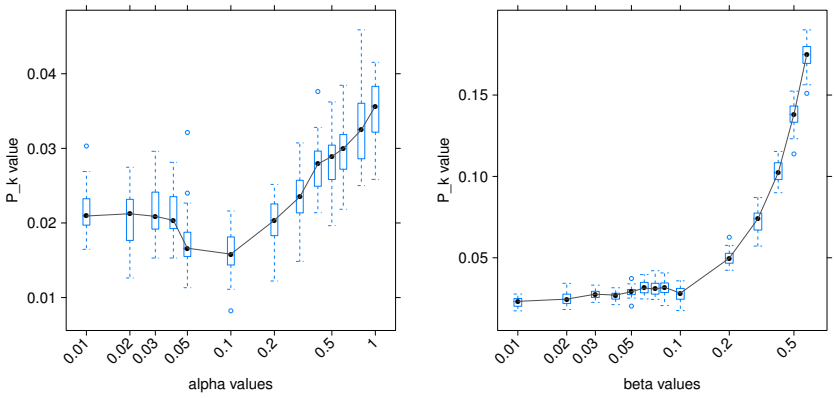
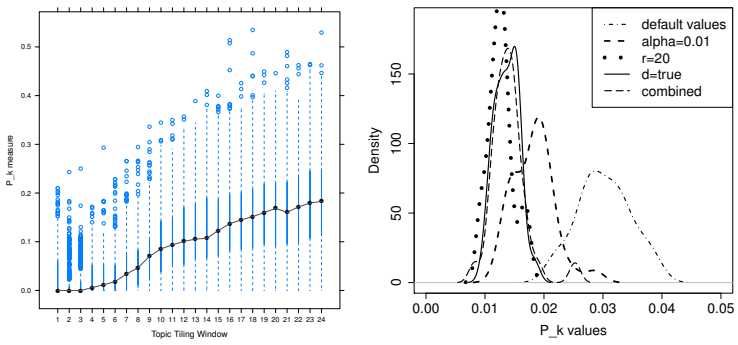


Figure 4.8: Box plot for different alpha (left) and beta (right) values with $m = 500$, $i = 100$, $T = 100$, $r = 1$ and $\beta = 0.1$ (left image) and $\alpha = 0.5$ (right image).

worse performance. Examining the variance, no patterns within the stable range emerge as observed in Figure 4.8.

Window Parameter w

The optimal window parameter has to be specified according to the documents that are segmented.



(a) window parameter (b) Error distributions

Figure 4.9: Figure a) represents the box plots for varying window parameter w with $m = 500$, $i = 100$, $T = 100$, $\alpha = 50/T$, $\beta = 0.1$, $r = 1$. The density of the error distribution for the system according to Table 4.2 is shown in Figure b).

Using the Choi corpus we observe that the window parameter can be increased to a size of 3 before the error rate increases. Since the segment sizes vary from 3-11 sentences we expect a decline for $w > 3$, which is confirmed by the results shown in Figure 4.9a.

4.5.3 Putting It All Together

Until this point, we have examined different parameters with respect to stability and error rates one at the time. Now, we combine what we have learned from this and strive at optimal system performance. Table 4.2 presents P_k error rates for the different systems. At this, we fixed the following parameters: $T = 100$, $m = 500$, $i = 100$, $\beta = 0.1$. We use 600 documents for the LDA model estimation and apply TopicTiling to the 100 remaining documents and repeat this 30 times with different random seeds.

Parameters	P_k	**error reduction**	σ^2	**variance reduction**
default	0.0302	0.00%	2.02e-5	0.00%
$\alpha = 0.1$	0.0183	39.53%	1.22e-5	39.77%
$r = 20$	0.0127	57.86%	4.65e-6	76.97%
$d = true$	0.0137	54.62%	3.99e-6	80.21%
combined	0.0141	53.45%	9.17e-6	54.55%

Table 4.2: Comparison of single parameter optimizations and combined parameter settings for TopicTiling. P_k averages and variance are computed over 30 runs, together with reductions relative to the default setting. Default: $\alpha = 0.5$, $r = 1$, $d = false$. combined: $\alpha = 0.1$, $r = 20$, $d = true$

We observe massive improvements for optimized single parameters. The α-tuning results in an error rate reduction of 39.77% in comparison to the default configurations. Using $r = 20$, the error rate is cut in less than half its original value. Also, for the mode mechanism ($d = true$) the error rate is halved but slightly worse than when using the repeated inference. Regarding the practice to assign the most frequent topic ID selected from every 50-100th iteration, we conclude that — at least in our application — a much smaller number of iterations suffices when taking assignments from all iterations. Here, allowing long inference periods to account for possible topic drifts seems not required. Using combined optimized parameters does not result to additional error decreases. We attribute the slight decline of the combined method in both in the error rate P_k and in the variance to complex parameter interactions that shall be examined in further work. In Figure 4.9b, we visualize these results in a density plot. It becomes clear that repeated inference leads to slightly better and more robust performance (higher peak) than the mode method. We attribute the difference to situations, where there are several highly probable topics in our sampling units, and by chance, the same one is picked for adjacent sentences that belong to different segments, resulting in failure to recognize the segmentation point. Since the differences are miniscule, only using the mode method might be more suitable for practical purposes since its computational cost is lower.

4.6 Comparison to Other Algorithms

In a last series of experiments, we compare the performance of TopicTiling with other TS algorithms on several datasets. All LDA models for these series are created using $T = 100$, $\alpha = 50/T$, $\beta = 0.01$, $m = 500$, $i = 100$.

4.6.1 Evaluation on the Choi Dataset

The evaluation uses the 10-fold CV setting as described in Section 4.4.3. For this dataset, no word filtering based on parts of speech was deemed necessary. The results for different parameter settings are listed in Table 4.3. Using only the window parameter without the mode

Parameters	**3-5**		**6-8**		**9-11**		**3-11**	
	P_k	WD	P_k	WD	P_k	WD	P_k	WD
d=false,w=1	2.71	3.00	3.64	4.14	5.90	7.05	3.81	4.32
d=true,w=1	3.71	4.16	1.97	2.23	2.42	2.92	2.00	2.30
d=false,w=2	1.46	1.51	1.05	1.20	1.13	1.31	1.00	1.15
d=true,w=2	**1.24**	**1.27**	**0.76**	**0.85**	**0.56**	**0.71**	**0.95**	**1.08**
d=false,w=5	2.78	3.04	1.71	2.11	4.47	4.76	3.80	4.46
d=true,w=5	2.34	2.65	1.17	1.35	4.39	4.56	3.20	3.54

Table 4.3: This table shows results for TopicTiling based on Choi's dataset with varying parameters.

($d = false$), the results demonstrate a significant error reduction with a window of 2 sentences. An impairment is observed when using a too large window (w=5) (cf. Section 4.5.2). We can also see that the mode method improves the results when using a window of 1, except for the documents with small segments ranging from 3-5 sentences. The lowest error rates are obtained with the mode method and a window size of 2.

As described in Section 4.4.2, the algorithm is also able to automatically estimate the number of segments using a threshold value (see Table 4.4). Here, the optimized parameters lead

Parameters	**3-5**		**6-8**		**9-11**		**3-11**	
	P_k	WD	P_k	WD	P_k	WD	P_k	WD
d=false,w=1	**2.39**	**2.45**	4.09	5.85	9.20	15.44	4.87	6.74
d=true,w=1	3.54	3.59	1.98	2.57	3.01	5.15	2.04	2.62
d=false,w=2	15.53	15.55	0.79	0.88	1.98	3.23	1.03	1.36
d=true,w=2	14.65	14.69	**0.62**	**0.62**	**0.67**	**0.88**	**0.66**	**0.78**
d=false,w=5	21.47	21.62	16.30	16.30	6.01	6.14	14.31	14.65
d=true,w=5	21.57	21.67	17.24	17.24	6.44	6.44	15.51	15.74

Table 4.4: Results for TopicTiling on Choi's dataset when estimating the number of segments automatically

to worse results for segments of length 3-5. This is caused by the smoothing effect of the window parameter, which leads to less detected boundaries. Nevertheless, the results of the other documents are comparable to the ones shown in Table 4.3. Some results (see segment length 6-8 and 3-11 with parameter d=true and w=2) are even better than the results when the number of segments is known beforehand. This is attributed to the remaining variance in the probabilistic inference computations. The threshold method can outperform the setup with a given number of segments, since not recognizing a segment produces less error in the measures than predicting a wrong segment. Table 4.5 presents a comparison of the performance of TopicTiling compared to different algorithms in the literature.

Method	**3-5**	**6-8**	**9-11**	**3-11**
TT (Choi, 2000)	44	43	48	46
C99 (Choi, 2000)	12	9	9	12
U00 (Utiyama and Isahara, 2001)	9	7	5	10
LCseg (Galley et al., 2003)	8.69			
F04 (Fragkou et al., 2004)	5.5	3.0	1.3	7.0
M09 (Misra et al., 2009)	2.2	2.3	4.1	2.3
STM (Du et al., 2013)	**1.0**	0.9	1.2	**0.6**
TopicTiling (d=true, w=2)	1.24	**0.76**	**0.56**	0.95

Table 4.5: This table presents the lowest P_k values for Choi's text segmentation dataset for various algorithms in the literature where the number of segments is known beforehand.

The results achieved with TopicTiling are far better than most current state-of-the-art results, except the STM. For these we observe significant improvements using a one sampled t-test with $\alpha = 0.05$. Whereas the STM method achieves better results for the segments 3-5 and 3-11, our method seems to work much better when the segments become longer.[24] With error rates below the 1% range, TS on the Choi dataset can be considered as solved. Since the dataset is comparatively easy, and test data has probably been seen during model training (cf. Section 4.4.3), we assess the performance of our algorithm on a second dataset.

4.6.2 Evaluation on Galley's WSJ Dataset

For the evaluation based on Galley's WSJ dataset we use the WSJ collection of the PTB to train the topic model. This collection consists of 2499 articles and is the same as used by Galley to create the text segmentation dataset. The evaluation generally leads to higher error rates than in the evaluation for the Choi dataset, as shown in Table 4.6.

This table shows results of the WSJ data when using all words of the documents for training a topic model and assigning topic IDs to new documents. Additionally, we report on results

[24]Due to the lack of missing results of the STM method for the more challenging WSJ dataset (see Section 4.6.2) we cannot state, which method in general performs best.

Parameters	All words		Filtered	
	P_k	*WD*	P_k	*WD*
d=false,w=1	37.31	43.20	37.01	43.26
d=true,w=1	35.31	41.27	33.52	39.86
d=false,w=2	22.76	28.69	21.35	27.28
d=true,w=2	21.79	27.35	19.75	25.42
d=false,w=5	14.29	19.89	12.90	18.87
d=true,w=5	**13.59**	**19.61**	**11.89**	**17.41**
d=false,w=10	14.08	22.60	14.09	22.22
d=true,w=10	13.61	21.00	13.48	20.59

Table 4.6: This table presents results for Galley's WSJ dataset using different parameters for TopicTiling for unfiltered documents (column 2-3) and for filtered documents using only verbs, nouns (proper and common) and adjectives (column 3-4).

using only nouns (proper and common), verbs and adjectives[25]. Considering the unfiltered results, we observe that performance benefits from using the mode assigned topic ID and a window larger than one. In case of the WSJ dataset, we find the optimal setting for the window parameter to be 5. As the test documents contain whole articles, which consist of at least 4 sentences, a larger window is advantageous here, yet a value of 10 is too large. Filtering the documents for parts of speech leads to $\sim$ 1% absolute error rate reduction, as can be seen in the last two columns of Table 4.6. Again, we observe that the mode assignment always leads to better results, gaining at least 0.6%. Especially the window size of 5 helps TopicTiling to decrease the error rate to a third of the value observed with d=false and w=1. Table 4.7 shows the results we achieve with the threshold-based estimation of segment boundaries for the unfiltered and filtered data.

In contrast to the results obtained with the Choi dataset (see Table 4.4), no decline occurs when using the threshold approach in combination with the window method. We attribute this due to the small segments and documents in the Choi dataset. Part-of-speech-based filtering is always advantageous over using all words here. In addition, a decrease of both error rates, P_k and *WD*, is detected when using the mode and using a larger window size. An improvement is even achieved for a window of size 10. This can be attributed to the fact that using small window sizes, too many boundaries are detected. As the window approach smooths the similarity scores, this results to less segmentation boundaries and improved results.

Table 4.8 presents the results of other algorithms, as published by Galley et al. (2003), in comparison to TopicTiling. Again, TopicTiling improves over the state of the art. The improvements with respect to LCseg are significant using a one-sample t-test with $\alpha = 0.05$.

[25] as identified by the TreeTagger (Schmid, 1995): `http://code.google.com/p/tt4j/`.

Parameters	All words		Filtered	
	P_k	WD	P_k	WD
d=false,w=1	53.07	72.78	52.63	72.66
d=true,w=1	53.42	74.12	51.84	72.57
d=false,w=2	46.68	65.01	44.81	63.09
d=true,w=2	46.08	64.41	43.54	61.18
d=false,w=5	30.68	43.73	28.31	40.36
d=true,w=5	28.29	38.90	26.96	36.98
d=false,w=10	19.93	32.98	18.29	29.29
d=true,w=10	**17.50**	**26.36**	**16.32**	**24.75**

Table 4.7: This table illustrates TopicTiling results for the WSJ dataset without providing the number of segments. Columns 2 and 3 present the results when using all words of the documents. Columns 4 and 5 show the results with part-of-speech-based filtering.

Method	P_k	WD
C99 (Choi, 2000)	19.61	26.42
U00 (Utiyama and Isahara, 2001)	15.18	21.54
LCseg (Galley et al., 2003)	12.21	18.25
TopicTiling (d=true,w=5)	**11.89**	**17.41**

Table 4.8: This table shows the best result of TopicTiling based on the WSJ dataset. Additionally, we present values for C99, U00 and LCseg as stated in (Galley et al., 2003).

4.7 Conclusion

This chapter demonstrated how the topic model LDA is used for text segmentation. We showed that replacing words in documents by topic IDs, as assigned by the Bayesian inference method of LDA, leads to better results based on Text Segmentation tasks. This technique is applied in the TT and C99 algorithms. Additionally, we introduced a simplified algorithm based on TT, called TopicTiling that outperforms the topic-based versions of TT and C99. In contrast to other TS algorithms using topic models (Misra et al., 2009; Sun et al., 2008; Du et al., 2013), the runtime of TopicTiling is linear in the number of sentences. This makes TopicTiling a fast algorithm with complexity of $O(n)$ (n denoting the number of sentences) as opposed to $O(n^2)$ of the dynamic programming approach as discussed by Fragkou et al. (2004).

During sweeping the parameter space of LDA and TopicTiling (see Section 4.5), we show that repeating the Bayesian inference several times and using the most frequently assigned topic IDs in the last iteration not only reduces the variance, but also improves overall results. We obtain almost equal performance when selecting the most frequent topic ID (mode) assigned per word across each inference step. Although the error rates are slightly higher in our experiments, this method is preferred, as the computational cost is much lower than repeating

the inference step several times. This method is not only applicable to Text Segmentation, but in all applications where performance crucially depends on stable topic ID assignments per token. Using Choi's dataset, only one algorithm can achieve slightly better results for short segments. However, based on Galley's WSJ dataset we show the best results in comparison to state-of-the-art algorithms.

TopicTiling has already been applied successfully in different experiment. Kaur and Mangat (2013) used the algorithm to segment text that was extracted automatically from images. In the article by Özmen et al. (2014) TopicTiling was figured out to be the best system when using it to segment unstructured e-learning course material into single topics.

Equipped with a highly reliable segmentation mechanism, TopicTiling can be applied as a writing aid to assist authors with feasible segmentation boundaries. This could be applied in an interactive manner by giving feedback about the coherence during the writing process. As the author is responsible for accepting such segmentation, the need for automatically determining the number of segments would be dispensable, and subject to tuning to the author's preferences.

Segmenting text into topically coherent parts provides the computer with information about related sentences. But no information about the meaning of a word is obtained using this segments. A method for representing the meaning of a word in order to compute similarities between words is subject of the next chapter.

Chapter 5

A Generic Framework for Computing Distributional Thesauri

dictionary, vocabulary, grammar, Thesaurus, glossary, lexicon, ontology, directory, encyclopedia, phonology

Similar terms for the term *thesaurus* from a distributional thesaurus described in this chapter

To understand the meaning of a *language element*, humans rely on contextual representations as described in Section 2.3. In addition to contextual representations, paradigmatic relations (similarities between *language elements*) deserve an important role. Here, we introduce a framework for computing similarities between *language elements*, resulting in a distributional thesaurus (DT) as described in Section 3.2.3. In contrast to existing approaches, we do not follow dense numeric vector space-based similarity computations but focus on a symbolic graph-based approach. For flexibility we split the computation of a DT into two parts. The first part is the context extractions, introduced in Section 2.4. This computation step transforms raw text into a tuple representation of *language elements* and *context features*. The second part is an efficient similarity computation, which is designed to scale to arbitrarily large corpora and uses Hadoop's MapReduce. As the similarity computation relies on a tuple-based representation, it can be performed in a language-independent way. We show the performance of different parameters and compare our method to standard approaches for computing DTs. This chapter is based on parts of the following publications: Biemann and Riedl (2013); Riedl and Biemann (2013a); Riedl et al. (2014c).

5.1 Introduction

With the advent of large text corpora and reasonably precise methods to automatically assign grammatical structure to sentences, it became possible to compute term similarities for a large vocabulary (Ruge, 1992). Lin (1998) computed a DT by comparing context features defined over grammatical dependencies with an appropriate similarity measure for all reasonably frequent words in a large collection of text (see Section 3.2.3 for more details). In order to evaluate these automatically computed word similarities, lexical resources are used. Entries in the DT consist of a ranked list of the globally most similar *language elements* (here: terms) per *language element* of interest. These similarities are dependent on the instantiation of the context feature as well as on the underlying text collection. We call them global similarities, as *language elements* can have different senses (e.g. jaguar can be both a car brand and an animal), but the different meanings are not separated but mixed within the similarities for a *language element*.

As described in Chapter 1, we target on a symbolic representation to operationalize distributional similarities following de Saussure (1959). A review on the connection of de Saussure's linguistic theories and distributional similarity was presented by Sahlgren (2006). While Sahlgren (2006) motivated vector-space approaches to model the meanings of words, we do not agree that "nouns are vectors, and adjectives are matrices" (Baroni and Zamparelli, 2010), although they can of course be represented in these or similar ways. Vector space representations are becoming increasingly successful in modeling natural language semantics, but vectors are typically too sparse and too highly dimensional to be used in their canonical form, and do not (naturally) encode relations beyond undifferentiated co-occurrence. We argue that there is no need to explicitly model non-existing relations, which would be zeros in the vector representation. Furthermore, our approach allows an interpretable context representation, which is important for reasoning decision in e.g. medical systems (cf. Watson Paths (Lally et al., 2014)). Additionally, a graph-based approach can be easily structured and extended to e.g. represent taxonomic or other relations. In many other approaches, the time for computing similarities for each *language elements* is very time-consuming and often disregarded. Here we present an approach, resulting in a DT that includes similarities for all *language elements*. This is accomplished with pruning strategies and Hadoop's MapReduce paradigm, which furthermore enables our method to scale to arbitrary amounts of data.

5.2 Related Work

A variety of approaches to compute DTs, have been proposed to tackle issues regarding size and runtime. The reduction of the feature space seems to be one possibility, but still requires the computation of such reduction (cf. Blei et al. (2003); Golub and Kahan (1965)). Other approaches use randomized indexing for storing counts or hashing functions to approximate counts and measures (Gorman and Curran, 2006; Goyal et al., 2010; Sahlgren, 2006). An-

other possibility is the usage of distributed processing like MapReduce. In (Pantel et al., 2009; Agirre et al., 2009), a DT is computed using MapReduce on 200 quad core nodes (for 5.2 billion sentences) respectively 2000 cores (1.6 Terawords), an amount of hardware only available to commercial search engines. Whereas Agirre et al. uses a χ^2 test to measure the information between *language elements* and *context features*, Pantel uses the Pointwise Mutual Information (PMI). Then, both approaches compute the cosine similarity to obtain the similarity between terms. Furthermore, Pantel describes an optimization for the calculation of the cosine similarity. Whereas Pantel and Lin (2002) describe a method for sense clustering, they also use a method to calculate similarities between terms. Here, they propose a pruning scheme similar to ours, but do not explicitly evaluate its effect. The evaluation of DTs has been performed in both extrinsic and intrinsic manners. Extrinsic evaluations have been performed using e.g. DTs for automatic set expansion (Pantel et al., 2009) or phrase polarity identification (Goyal and Daumé, 2011).

Lin (1997, 1998) introduced two intrinsic measures using WordNet (Miller, 1995) and Roget's Thesaurus (Berry, 1962). Using WordNet, he defines *context features* (synsets a word occurs in WordNet or subsets when using Roget's Thesaurus) and then builds a gold standard thesaurus using a similarity measure. Then he evaluates his generated DT with respect to the gold standard thesauri. Weeds (2003) evaluates various similarity measures based on 1000 frequent and 1000 infrequent words. Curran (2004) created a gold standard thesaurus by manually extracting entries from several English thesauri for 70 words. His automatically generated DTs are evaluated against this gold standard thesaurus using several measures. We use his measure during our evaluation as well as a measure based on WordNet.

The WordNet-based measure, which is introduced in Section 5.4.2, has also been applied by Padró et al. (2014). Whereas we perform the evaluation mainly based on nouns, they evaluate based on verbs. They apply different pruning and ranking strategies for computing similarities and tune them in order to detect the best parameters. We compare our method to their approach in Section 5.7.6.

5.3 Computing Distributional Similarities

The computation of the similarities between *language elements* relies on tuples extracted by the context extraction method as introduced in Section 2.4. We compute similarities between all *language elements*, which results in a distributional thesaurus (DT) and is implemented based on Apache Hadoop's MapReduce framework[26]. This framework allows parallel processing of large amounts of textual data. It is based on the principle developed by Dean and Ghemawat (2004) and uses two steps, namely Map and Reduce. The Map step converts input text to key-value pairs, sorted by key. Then, the Reduce step operates on all values that have the same key,

[26] `http://hadoop.apache.org`

producing again a data table with a key. As these steps do not require a global information flow, many Map and Reduce steps can be executed in parallel, allowing the system to scale to huge amounts of data. Further, Apache Pig[27] is used, a query language similar to SQL that allows us to perform database joins, sorting and limit operations on Hadoop data tables.

The workflow of the DT is illustrated in Figure 5.1 and starts with a collection of sentences as input. The first step is the observation extraction and holing operation (see Section 2.4),

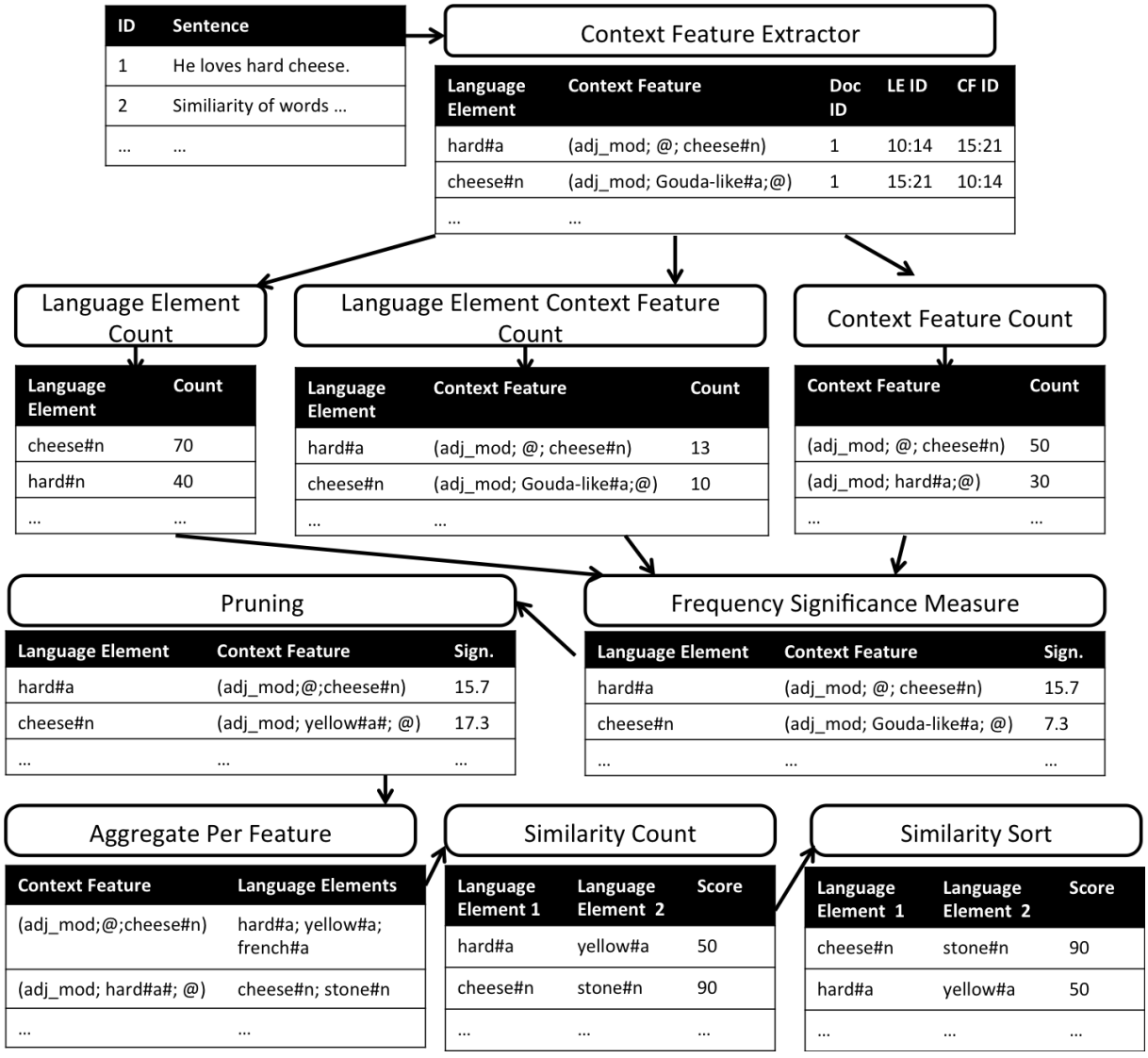


Figure 5.1: This figure illustrates the workflow for our similarity computation between *language elements* using Hadoop's MapReduce.

which converts the documents into a tuple representation of *language elements* and *context features* including identifiers for the *language element* (LE ID), the *context feature* (CF ID) and

[27] http://pig.apache.org/

the document (Doc ID). This information is necessary to compute correct counts of *language elements* and *context features*. In the example of the workflow, a dependency parser is used for the holing operation (see example *b)* in Section 2.4.2) using lemmatized words and their POS tags.

According to our definition in Section 2.5 we use $x \in X$ to represent a *language element* and $y \in Y$ for a *context feature*. In the next step, we aggregate all equal tuples and compute the frequencies of *language elements* $|\langle x, * \rangle|$, *context features* $|\langle *, y \rangle|$ and co-occurrence of *language elements* and *context features* $|\langle x, y \rangle|$. These frequencies are required to compute the significance measures of each tuple. Then, these measures are used to rank important *context features* for each *language element*. We perform experiments based on the information measures PMI, LMI and LL as introduced in Section 2.7.1. Additionally, we apply the frequency of the co-occurrence of *language elements* and *context features*. Whereas this is not a significance measure, the frequency showed the best results in an intrinsic evaluating of DTs performed by Padró et al. (2014).

In order to compute these measures, the tables produced by *Language Element Count* and *Language Element - Context Feature Count* are joined to the table holding frequencies of term-feature pairs using an Apache Pig script. A different approach for computing the term co-occurrences is described by Lin and Dyer (2010). He proposes to load the single frequencies into memory to avoid the join operation and to speed up the overall computation. While this works for a limited (albeit large) vocabulary of terms when carefully tuning the number of Mappers per computation node, it imposes a severe limitation on the number of (arbitrary complex and productive) *context features*. This is the reason for not applying this design pattern.

In the next step we build a graph by adding all *language elements* and *context features* as nodes. Each *language element* x has the *context features* Y_x with $\forall y \in Y_x | y \in Y \wedge |\langle x, y \rangle| > 0$. We also obtain all *language elements* for a *context feature* with X_y with $\forall x \in X_y | x \in X \wedge |\langle x, y \rangle| > 0$. Then we connect each term with the context features it co-occurs, using the tuple information and use the significance measure as edge weight $s(\langle x, y \rangle) = \textit{significance}$. For an efficient computation of DTs, we introduce several pruning parameters, as shown in Table 5.1 and explained next.

In the *Pruning* step, we remove *language elements* with frequencies below w ($|\langle x, * \rangle| < w$) and context features below f ($|\langle *, y \rangle| < f$). Additionally, we provide a parameter to delete edges between *language elements* and *context features*, which occur less than wf ($|\langle x, y \rangle| < wf$). We use the parameter *sig* to prune edges according to the significance score ($s(\langle x, y \rangle) < sig$). It is advised to remove edges between *language elements* and *context features* that have a negative correlated significance score. Removing these edges is similar to the idea of using the positive PMI, which proves to be beneficial as shown by e.g. Levy and Goldberg (2014a). This measure only uses positive PMI values and assigns a PMI value of zero to negative scores. According to the power-law distribution (see Section 2.6), *context features*, which co-occur

Parameter	Description	Default value
w	minimum *language element* count	1
f	minimum *context feature* count	1
wf	minimum *language element - context feature* count	1
sig	minimum significance score	0
$wpfmax$	maximal number of terms for a *context feature*	1000
$wpfmin$	minimal number of terms for a *context feature*	1
p	use the top p ranked *context features* per term	1000
l	extract for each term the most l similar terms	200

Table 5.1: List of parameters including some default values for computing a DT with our framework.

with too many *language elements*, tend to be very general and do not contribute when compucting similarities between *language elements* (cf. Kilgarriff et al., 2004; Goyal et al., 2010). Thus, we delete all *context features* in the graph that occur with more than *wpfmax language elements* ($|X_y| > wpfmax$). Additionally, we remove features that occur with few *language elements* with the parameter *wpfmin* ($|X_y| < wpfmin$). Still, the number of *context features* per *language element* can be too high for a fast computation of similarities between *language elements*. To further decrease the number of *context features*, we keep only the p highest ranked *context features* for each *language element*. For each *language element*, we first rank the *context features* according to their significance score and then remove all remaining edges. It is sufficient to keep only the p most salient features per term, obtained with $highest(x, p)$ (see Section 2.5), as features of low saliency generally should not contribute much to the similarity of *language elements* and could lead to spurious similarity scores. These pruning steps make our approach feasible for large amounts of text. The influence of these parameters on the quality of the DT is examined in Section 5.7.1. The set of all pruned tuples T_{pruned} is then obtained with:

$$\{\ \forall \langle x, y\rangle \in T_{pruned} \mid y \in Y_x \wedge\ \langle x, y\rangle \in\ highest(Y_x, p) \wedge \\ |\langle x, y\rangle| >\ wf\ \wedge |\langle *, y\rangle| >\ f\ \wedge |\langle x, *\rangle| >\ w \wedge \\ s(\langle x, y\rangle) >\ sig\ \wedge |X_y| >\ wpfmin \wedge\ |X_y|\ <\ wpfmax\ \}.$$

We define all remaining *language elements* for a given *context feature* y with $X_{y,pruned}$ and all remaining *context features* for a *language element* x with $Y_{x,pruned}$.

Afterwards, we aggregate all terms by their features (*Aggregate Per Feature*). This allows us to compute similarity scores between all terms that share at least one feature (*Similarity Count*). Within the *Similarity Count* step, we generate all combinations of the aggregated terms and record only the information that these two terms occur together with one feature. If this step has been performed for all aggregated terms, the counts for the occurrences of two

terms need to be summed up, which is the number of pruned features two terms share. This constraint makes this approach scalable to larger data, as it is not necessary to know the full list of features for a term pair at any time. In Section 5.7.4 we show that these simplifications do not impair the quality of the obtained DT, especially when using large corpora.

Although our similarity score is not a metric, as we do not retrieve scores between 0 and 1, we prove its validity for ranking similar terms. We can show that our similarity score is proportional to the Jaccard metric (Jaccard, 1912) when we normalize the *context feature* overlap of two *language elements* by the frequency of the union of their features. Thus, the Jaccard similarity between two terms can transformed to:

$$\begin{aligned} jaccard(x_i, x_j) &= \frac{|Y_{x_i,pruned} \cap Y_{x_j,pruned}|}{|Y_{x_i,pruned} \cup Y_{x_j,pruned}|} \\ &\approx \frac{|Y_{x_i,pruned} \cap Y_{x_j,pruned}|}{2 \cdot p - |Y_{x_i,pruned} \cap Y_{x_j,pruned}|} \\ &\propto |Y_{x_i,pruned} \cap Y_{x_j,pruned}| \end{aligned} \tag{5.1}$$

As the parameter p is a constant, this ranking is only dependent on the overlap of the two terms. Thus, our simple similarity score, which counts only the feature overlap of two terms, is proportional to the Jaccard similarity.

The last step sorts the list by terms and by descending score. To reduce the size of the output, only the l most similar terms per entry are kept. The overall computation results in second order (paradigmatic) similarities including the similarity scores. Furthermore, the first order (syntagmatic) significant pairs $\langle x, y \rangle$ with their significance scores are helpful as features for further tasks, e.g. contextualization (Riedl and Biemann, 2013b) and lexical substitution (see Section 8.2).[28]

To evaluate the performance of the computational approach for DTs, the next step is to introduce evaluation methods that are suited for measuring the quality of DTs.

5.4 Evaluation of Distributional Thesauri

Evaluating the performance of unsupervised methods, which learn structure from data is always challenging. Data used to validate such approaches, called gold standard, are mostly generated manually. However, these resources mostly do not fit perfectly to the structure extracted by the method. This does also apply to methods, computing distributional similarities between *language elements*. These methods are heavily dependent on the input data and it is generally impossible to generate a manually created gold standard that covers the full vocab-

[28] The implementation of the framework is available via the JoBimText project as open-source software under the ASL 2.0 for download: `http://sf.net/p/jobimtext/`. Models can be viewed directly using the Web demo available at: `http://maggie.lt.informatik.tu-darmstadt.de/jobimviz/` (Ruppert et al., 2015a).

ulary found in text. Furthermore, language changes over time (Mitra et al., 2014) and thus, resources need to be consistently updated. This is time-consuming and often associated with high costs.

Nevertheless, methods and datasets have been introduced to evaluate distributional semantic models. Rubenstein and Goodenough (1965) proposed one of the first datasets for evaluating associations between terms. This dataset consists of 65 word pairs with scores from 0 to 4 indicating the similarity between the pairs. Miller and Charles (1991) could show that human judgment is highly correlated to similarities achieved by using context representation to compute similarity between terms. Similar but slightly larger datasets have been introduced with WordSim353 (Finkelstein et al., 2001) and MC-30 (Miller and Charles, 1991). Instead of word similarity, these datasets measure how associated two words are (Hill et al., 2014). Thus, word pairs do not only achieve high scores if they are similar, also called near-synonymy relation (e.g. car and automobile), but also if they are related (e.g. gas and car). LSA and topic models like LDA (see Sections 3.1.2 and 3.1.3) cluster words that are related and similar, leading to a more topical word similarity. However, a DT should be a resource where only similar words are grouped together. Additionally, these datasets are annotated with scores within specific ranges, which might be too coarse-grained to be useful for evaluating distributional similarity. Hill et al. (2014) tried to overcome this problem and provide more instances. However, using such datasets requires to provide a score for all word pairs even if they are topically unrelated.

Another evaluation setting for DTs is comparing them against manually created thesauri (Curran, 2004; Lin, 1997, 1998). Curran (2004) created a gold standard thesaurus by manually extracting entries for 70 words from several English thesauri. His automatically generated DTs are evaluated against this gold standard thesaurus using several measures. Whereas such an evaluation should be valid, there, again, are some drawbacks: most manually created thesauri are not available online and thus the gold standard has to be created manually. Additionally, new senses for a word might not be contained and some old senses are still present. An automatic approach could be performed using WordNet (Miller, 1995). Lin (1997, 1998) introduced an evaluation method using WordNet and Roget's Thesaurus (Berry, 1962). He defines *context features* (WordNet: synsets a word occurs when using WordNet and subsets a word is contained when using Roget's Thesaurus) and then builds a gold standard thesaurus using a similarity measure. Then, he compares his generated DT with respect to the gold standard thesauri. Weeds et al. (2004) evaluate various similarity measures based on 1000 frequent and 1000 infrequent words. In addition to the intrinsic evaluation settings, extrinsic evaluations have been performed e.g. for automatic set expansion (Pantel et al., 2009) or phrase polarity identification (Goyal and Daumé, 2011).

In this chapter, we employ the candidates also used by Weeds et al. (2004), which are nouns. We introduce two different evaluation methods to evaluate DTs, which could be performed for both English and German DTs. We perform evaluations against manually created thesauri and

against taxonomies like WordNet. To adapt the evaluation to e.g. German, we replace WordNet with GermaNet (Hamp and Feldweg, 1997) (see Chapter 6).

5.4.1 Evaluation With Manually Created Thesauri

Focusing on evaluating against manually created thesauri, a gold standard needs to be generated. We consider the 1000 infrequent and 1000 frequent nouns proposed by Weeds et al. (2004) for the evaluation. For each of these nouns, we extract similar and related terms from several thesauri to increase the coverage of the inventory. We extract the similar and related terms from Roget's 1911 thesaurus[29], Moby Thesaurus[30], Merriam Webster's Thesaurus[31], the Big Huge Thesaurus[32] and the OpenOffice Thesaurus[33]. Results are reported using the inverse ranking measure, the precision at 1 (P@1) and 5 (P@5) (Curran, 2002) and the generalized average precision (GAP) (Kishida, 2005; Thater et al., 2009).

For computing the various scores, we extract for each noun the l most similar terms from the DT and order them by their similarity score. Then these terms are given a rank from 1 to l. As an explanation we give an example for the term *colour* considering its ten most similar terms ($l = 10$), presented in Table 5.2. With $in\_gold(r)$ we test whether the term ranked

word	score	rank	in_gold	inverse rank	P@k
color	398.0	1	1	1/1	1/1
hue	238.0	2	0	0/2	1/2
tint	209.0	3	0	0/3	1/3
colouring	146.0	4	1	1/4	2/4
coloration	139.0	5	1	1/5	3/5
coloring	130.0	6	1	1/6	4/6
texture	122.0	7	0	0/7	4/7
red	113.0	8	0	0/8	4/8
complexion	111.0	9	0	0/9	4/9
brown	104.0	10	0	0/10	4/10
				SUM 1.667	

Table 5.2: This table gives an example for inverse ranking for the top 10 similar terms for *colour* and the $P@k$.

at position r is contained in the gold thesaurus. If it is contained, the function returns 1; otherwise it returns 0. Using this information, we can compute the precision at a specific rank

[29] We use the API from Jarmasz and Szpakowicz (2001).

[30] `http://icon.shef.ac.uk/Moby/`

[31] `http://www.dictionaryapi.com`

[32] `http://words.bighugelabs.com/api.php`

[33] The thesaurus is available at: `http://extensions.services.openoffice.org/en/dictionaries` and can be accessed via the open source tool Sinomini `http://sinonimi.sourceforge.net/`.

called *P@k*. To compute the score we sum up the *in_gold* values up to the rank k and divide the sum by the rank as shown in the following equation:

$$P@k = \frac{\sum_{r=1}^{k} \text{in\_gold}(r)}{k}. \tag{5.2}$$

Considering the example, the precision at rank one (*P*@1) is 1.0 and *P*@5 achieves a score of 3/5 = 0.6. In order to calculate the inverse rank for a term, we first compute the inverse rank of each similar term, which is assembled by dividing the *in_gold* value by its rank as shown in the column *inverse rank*. Summing up all inverse ranking scores we obtain a value of 1.667 for the term *colour*. The overall score is the sum of all inverse ranks divided by the number of words.

Additionally, we use the generalized average precision (GAP), which is commonly used for evaluating rankings. GAP works slightly differently from the inverse ranking and considers the number of correct answers at each ranking position. It is computed using the ratio of the ranking performed to the perfect ranking giving the same terms for a specific targeted term. The numerator is the sum of *P@k* scores multiplied by the *in_gold*(r) status at rank r and the perfect ranking is used for normalization. As *in_gold*(r) does not contain any fractions, the perfect ranking is just the number all the l terms, which occur in the manually created thesauri. Thus, the GAP is computed using the following equation:

$$GAP = \frac{\sum_{k=1}^{l} \text{in\_gold}(k) P@k}{\sum_{k=1}^{l} \text{in\_gold(k)}}. \tag{5.3}$$

Based on our example with $l = 10$, we achieve a GAP score of 0.692[34]. The GAP measure is a standard evaluation measure for ranking tasks. However, it has not been for evaluating DTs.

5.4.2 Evaluation Against WordNet

The second approach uses a taxonomy. Whereas in this chapter, we apply WordNet 3.1, WordNet could be easily replaced with GermaNet, which is used for evaluating German DTs in Chapter 6. We compute the similarity between two terms by applying the WordNet Path measure (Pedersen et al., 2004). The Path measure is the reciprocal distance between two terms t_1 and t_2 following the is-a relation path and adding +1 in order to have a similarity of one if two terms are within the same synset and have a path distance of zero:

$$path\_measure(t_1, t_2) = \frac{1}{min(isa\_path(t_1, t_2)) + 1}. \tag{5.4}$$

[34]The GAP is computed as follows: (1/1+2/4+3/5+4/6)/(1+1+1+1)=2.767/4=0.692

The highest possible score is one if two terms share a synset. Again, candidate words are used, which should also be contained in the taxonomy.[35] For each of the candidates the top l entries within the DT are extracted.[36] Then the average path score is computed for each candidate term $c_i \in C$. We use the average of the averaged scores of the similarity between c_i and its N most similar terms $t_i \in top(DT(t), N)$ as a score indicating the "similarity" of the DT to the taxonomy:

$$\text{wordnet\_path\_score(C)} = \frac{1}{|C|} \sum_{c_i \in C} \sum_{t_i \in top(DT(c_i), N))} \frac{path\_measure(c_i, t_i)}{N} \tag{5.5}$$

The example in Table 5.3 shows the top ten-ranked similar terms for the term *colour*. As *color* and *colour* share the same synset, the minimal path is zero resulting in a path score of one. The average score for the term *colour* using the WordNet-based Path measure is then

word	score	minimal path	path score
color	398.0	0	1/1
hue	238.0	3	1/3
tint	209.0	1	1/2
colouring	146.0	0	1/1
coloration	139.0	0	1/1
coloring	130.0	0	1/1
texture	122.0	2	1/3
red	113.0	2	1/3
complexion	111.0	1	1/2
brown	104.0	2	1/3
Average			0.633

Table 5.3: Example for WordNet Path scores for the term *colour*.

0.633.

Both evaluation methods become more complex when applying holing operations, which add POS tags to the terms (i.e. the dependency parse holing operation in Section 2.4.2). WordNet and GermaNet do not contain relations between two terms of different POS. Furthermore, lexical resources in general comprise only of base forms of words. However, when computing distributional similarities, we observe similarities also among terms with different POS tags and varying word forms. Within the evaluation, we remove terms from the DT with different POS tags. Examinations of the impact of different strategies have shown only marginal variations in the evaluation scores.

[35] The list of candidate terms for English, which are the ones introduced by Weeds (2003), is reduced by the words *goods* and *supplies*. These words only occur as base forms in WordNet.

[36] If a similar term is not contained in WordNet, it is neglected and we add the next entry to the top N set of similar entries, as we cannot judge whether the term is correct or incorrect.

5.5 Experimental Setting

We evaluate the different parameters for building DTs using the two previously introduced evaluation measures. The evaluation based on nouns is performed using 1000 frequent and 1000 infrequent nouns from the British National Corpus (BNC), previously introduced by Weeds (2003). In Section 5.7.6 we also show evaluation results for DTs considering verbs. We use the verbs provided by Padró et al. (2014), which consists of 1509 low frequent, 1227 mid frequent and 1218 high frequent verbs extracted from the BNC.

5.6 Corpora

We perform the computation of DTs using different corpora. Most experiments are performed with a newspaper corpus, which consists of 105 million unique sentences (about two gigawords). This corpus is compiled from the freely available Leipzig Corpora Collection (LCC)[37] and from the Gigaword corpus (Parker et al., 2011).[38] As newspaper text covers a wide range of topics of the open domain, we expect a better quality than just using Web text, as newspaper articles are mostly written by journalists and reviewed.

To compare the performance of DTs computed on different corpora, we use a dump of the English Wikipedia[39] from 2011, which consists of 35 million sentences after cleaning. Wikipedia is an encyclopedia, which offers easy access to its data. Even though it is biased to some specific topics (e.g. cities and persons), it covers also a wide spectrum of the open and specific text domains.

Additionally, we use the BNC[40], which contains about 5 million sentence. It was composed of different sources of written English text, which should reflect the English language. Whereas this corpus is rather small, it is expected to have a high quality, as the text has been manually selected.

Furthermore, we use the Google books corpus, which was parsed and provided as syntactic n-grams by Goldberg and Orwant (2013) and consists of 17.6 billion sentences. Some errors within these resources are partially caused by the optical character recognition (OCR) process (see Riedl et al. (2014c)). However, due to its size, these errors should not influence the results too much when used for building DTs.

During the experiments, we also use fractions of the Wikipedia, newspaper and BNC corpus. In order to achieve similar-sized corpora, we downsample the corpora by randomly selecting the same amount of sentences for each corpus. We downsample the corpora to contain

[37] Leipzig Corpora Collection, `http://corpora.uni-leipzig.de`, Richter et al. (2006).

[38] Biemann and Riedl (2013) report a size of 120M sentences, which is the total size of sentences without any pruning. But not all sentences could be parsed due to their length (too short or too long) and thus are not used for the DT computation.

[39] `https://www.wikipedia.org/`

[40] `http://www.natcorp.ox.ac.uk/`

100k, 1M, 10M sentences, except for the BNC, which only contains 5M sentences and is thus only downsampled to 100k and 1M sentences.

Next, we show the results using these corpora and apply the previously introduced evaluation methods.

5.7 Results for the Evaluation of Distributional Thesauri

This section highlights different aspects of the computation of DTs. The first section examines the influence of different parameters of our framework and uses a subset of the newspaper corpus. Section 5.7.2 demonstrates the effect of using different context feature representations. Furthermore, we also show the influence of the corpus size as well as the various significance measures introduced for ranking context features (see Section 2.7.1). In Section 5.7.3, we present the performance of our DTs evaluated against WordNet and a manually created thesaurus, as introduced in Section 5.4. The performance of our framework is compared against other methods in Section 5.7.4. For this, we select two standard approaches and also compare the performance of our method to word embeddings. The selection of the corpora for computing a DT is responsible both for the quality of a DT and for the domain it covers. For this, we perform an evaluation using different corpora for computing DTs in Section 5.7.5. In the last evaluation (see Section 5.7.6) we evaluate the quality of verbs in the DTs.

5.7.1 Tuning the Pruning

In an initial exploration, 10 million randomly sampled sentences from the newspaper corpus are used to compute DTs for different parameters. As holing operation, we use collapsed dependency parses[41] and lemmatized words, which is exemplified in Enumeration *b)* in Section 2.4. During our experiments, we use the default parameters shown in Table 5.1 in Section 5.3. These parameters do not perform any filtering on *language elements* ($w = 1$), on *context features* ($f = 1$), on co-occurrences of *language elements* and *context features* ($wf = 1$) and only consider positive significance scores ($s >= 0$). The number of *context features* per *language element* is fixed to the range of 1 – 1000 ($wpfmax = 1000$ and $wpfmin = 1$). We perform the evaluation based on nouns and apply the WordNet Path measure, using WordNet 3.1 (see Section 5.4.2).[42]

First, we show the impact of different significance measures and the top ranked number of *context features* per term, which is specified by parameter p. This parameter has influence to the run-time of the DT computation and the file size of the intermediate and final outputs. We

[41] We use the Stanford parser `http://nlp.stanford.edu/software/lex-parser.shtml` (de Marneffe et al., 2006)

[42] In (Biemann and Riedl, 2013) we used WordNet 3.0 and applied the pruning parameters in a different order. Whereas the recommendation for the parameter choices remains the same, the results reported in this work are slightly higher.

present results for frequent and infrequent nouns separately in Tables 5.4 and 5.5. The results are averaged path scores for the 1000 nouns for the top 5 and 10 entries extracted from our DT for different significance measurements and for different settings of p.

According to the results in Table 5.4, we observe that scores for average path similarities over the top 5 terms are consistently higher than for the top 10 terms. This indicates that the ranking is valid with respect to semantic closeness.

Top	Sign.	max number of *context features* p					
words	Meas.	10	100	500	1000	5000	10000
top 10	Freq	0.1855	0.2762	0.2849	0.2810	0.2644	0.2503
top 10	PMI	0.0114	0.0556	0.1672	0.2089	0.2520	0.2436
top 10	LMI	0.1846	0.2742	0.2857	0.2847	0.2659	0.2568
top 10	LL	0.1789	0.2741	0.2856	0.2856	0.2671	0.2555
top 5	Freq	0.2104	0.3136	0.3216	0.3135	0.2915	0.2752
top 5	PMI	0.0166	0.0796	0.1829	0.2303	0.2776	0.2693
top 5	LMI	0.2085	0.3118	0.3220	0.3179	0.2959	0.2864
top 5	LL	0.1997	0.3092	0.3232	0.3207	0.3007	0.2852

Table 5.4: WordNet Path Scores for 1000 frequent nouns for DTs computed on 10 million sentences using a collapsed dependency parse holing operation.

Inspecting the scores for the different significance measures, we observe that the PMI measure does not play well with our pruning scheme regulated by the p parameter: while the other three measures yield very similar scores, PMI produces clearly inferior results. This confirms previous observations that PMI over-estimates *context features* with low word counts: These *context features* might characterize the terms extremely well, but are too sparse to serve as a basis for the computation of second-order similarity (cf. Bordag, 2008).

For all significance measures, except the PMI measure, the scores consistently decline with values above 1000 for the parameter p. As we treat each context feature equally for the similarity computation, rare context features are becoming more relevant for the similarity computation using the LMI, LL and Freq measure. This is confirmed by the results with p values above 1000: the evaluation scores for DTs using PMI are getting closer to the ones performed with the other measures. Thus, using too many context features for the similarity computation, the impact of the significance measure becomes mostly irrelevant.

The performance of the simple frequency measure achieves the highest scores with small values of p and is very close to the more sophisticated LMI and LL measures for higher p values. As we prune very frequent features with the *wpfmin* parameter, the frequency serves well for retrieving the relevant *context features* for each *language element*. The results achieved for the infrequent nouns (see Table 5.5) show a similar trend, but as expected the scores are much lower. This is partially due to the terms in the given noun list that are not contained in the DT. Furthermore, due to the lack of overall data for these terms, we obtain less reliable

similarities. A further reason is the incomplete WordNet coverage for senses that are dominant in our collection. In an evaluation where we used WordNet 3.0 (Biemann and Riedl, 2013), we discovered during data analysis that e.g. the term *anime* has two rather rare synsets: "a hard copal derived from an African tree" and "any of various resins or oleoresins". An entry for *anime* in the sense of the Japanese animation movie is missing. The entries of the DT using LMI and $p = 500$ contained "novel, music, manga, comic, cartoon, book, film, shows, sci-fi", which all received a low score. In WordNet 3.1 this sense is added, but as language evolves and lexical resources tend to be incomplete, still senses are missing. For instance the term *app*, referring to an application mostly for mobile phones, is not contained at all, which results to low similarity scores for the term *application*.

Top	Sign.	max number of *context features* p					
words	Meas.	10	100	500	1000	5000	10000
top10	Freq	0.1518	0.2100	0.2145	0.2126	0.2003	0.1922
top10	PMI	0.0035	0.1174	0.1854	0.1875	0.1877	0.1865
top10	LMI	0.1499	0.2065	0.2169	0.2133	0.2006	0.1943
top10	LL	0.1483	0.2051	0.2173	0.2134	0.2001	0.1937
top5	Freq	0.1706	0.2380	0.2423	0.2388	0.2235	0.2115
top5	PMI	0.0065	0.1333	0.2098	0.2112	0.2095	0.2067
top5	LMI	0.1683	0.2332	0.2459	0.2406	0.2211	0.2143
top5	LL	0.1640	0.2331	0.2458	0.2425	0.2221	0.2143

Table 5.5: WordNet Path scores for 1000 infrequent nouns for DTs computed on 10 million sentences using a collapsed dependency parse holing operation.

Although some senses and terms are missing in the lexical resource, we achieve a valid ranking for the infrequent nouns as the top 5 ranked terms score higher than the top 10 extracted words. Comparing the best results regarding parameter p, for both the infrequent and frequent nouns, there seems to be an optimal value for p. Considering more *context features* apparently does not improve the similarity and the highest values are obtained for p between 500 and 1000 in this experiment for LMI, LL and Freq. For infrequent terms, the difference between PMI and the other measures is much less pronounced. Yet we can safely conclude from these experiments that PMI is not the optimal measure in our setup. Whereas in most of the cases LL achieves the best performance, we use LMI for the following experiment as it seems to generalize better for improper p values and the differences to the results using LL are marginal. Furthermore, the formula for the LMI is simpler and can be computed much faster. Whereas the frequency achieves surprisingly good results, the results for the LMI and LL measure are mostly higher for the optimal choice for p.

It seems obvious that the parameter p is dependent on the corpus size used for computing a DT. We show correlation between the two parameters by performing an evaluation using different values of p in conjunction with different sized corpora. For this, we select the LMI

significance measure and randomly sample the full newspaper corpus to sizes of 100k, 1M, 10M and 105M sentences. The results are illustrated in Figure 5.2.

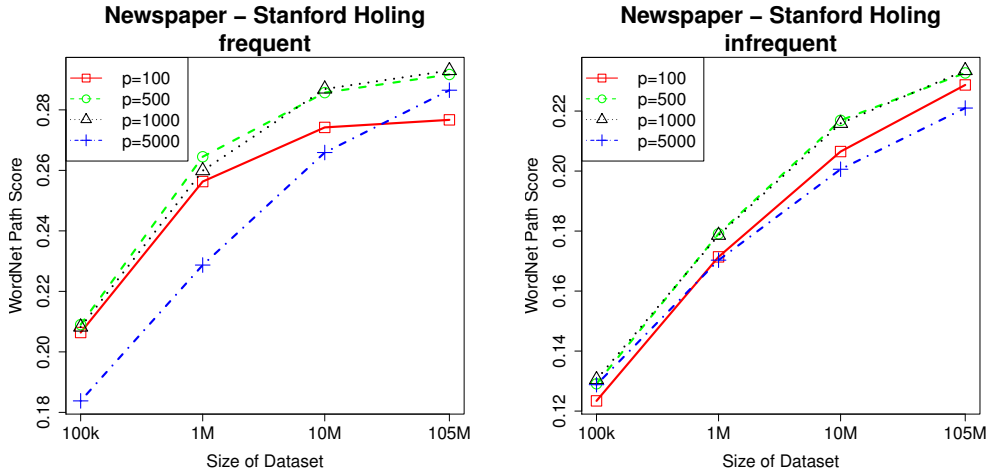


Figure 5.2: Different values of p for the LMI measure considering different corpus sizes. The evaluation is performed based on the top 10 most similar words using the WordNet Path evaluation measure.

As a general trend, larger corpora call for a larger value of parameter p. Considering the evaluation based on the frequent nouns (see left graph in Figure 5.2), we observe a similar performance for the DTs computed with 100k sentences for values of p in the range from 100 to 1000. Using 1M sentences we achieve the best results with $p = 500$ but based on a DT computed on 10M sentences, the results are close together for $p = \{500, 1000\}$. For the full corpus, we yield the best results for $p = 1000$. But still the results with p equals 500 are not much lower. Whereas for smaller corpora a small value of p seems to be sufficient, a larger value of p is advantageous for larger corpora. Computations with very high values do not seem to be viable, as can be seen for results with $p = 5000$. In general we observe that the scores increase with all parameters when using a larger corpora.

For the results of the infrequent nouns, which are presented in the right graph in Figure 5.2, a parameter in the range of $p = \{500, 1000\}$ accomplishes the best results. Remarkably, for the complete corpus, using $p = 100$ results to a better performance than for $p = 5000$. As infrequent nouns have less significant *context features* than frequent nouns, using high values for the parameter p does not filter less meaningful features. Thus, *context features* are introduced, which generate similarities to non-related terms. This can be observed as the score for the full corpus using $p = 5000$ is lowest. According to these experiments, we recommend choosing p in the range between 100 – 1000 for smaller corpora and 500 – 1000 for larger corpora.

In order to give some insight into the quality of the similarities obtained for different sized corpora, we show the most similar *language elements* for DTs that are computed using the LMI measure with $p = 1000$. We selected the frequent noun *beer* and for the rather infrequent noun

ion and show the similarities in Table 5.6 and 5.7. Additionally, we provide the WordNet Path scores in fractional notation, where $1/x$ indicates a path length of $x + 1$ between the selected *language element* and its similar *language elements*.

Language element: **beer**

rank	100k freq:	WP 74	1M freq:	WP 727	10M freq:	WP 6386	105M freq:	WP 30758
1	food	1/5	wine	1/4	brew	1/2	wine	1/4
2	play	1/12	drink	1/4	wine	1/4	drink	1/4
3	drug	1/5	food	1/5	beverage	1/4	coffee	1/5
4	wine	1/4	coffee	1/5	coffee	1/5	beverage	1/4
5	water	1/6	milk	1/5	vodka	1/5	lager	1/2
6	money	1/13	bottle	1/14	drink	1/4	soda	1/6
7	product	1/10	product	1/10	ale	1/2	tea	1/5
8	book	1/13	tea	1/5	cocktail	1/5	ale	1/2
9	vote	1/14	fuel	1/7	soda	1/6	liquor	1/4
10	card	1/10	cigarette	1/7	food	1/5	vodka	1/5
Ø		0.13		0.17		0.25		0.29

Table 5.6: This tables presents DT entries, computed with a *Stanford holing operation*, for the frequent noun *beer* with WordNet Path scores (WP). We show results for different corpus sizes from 100K sentences up to 105M sentences

It is apparent that for a frequent *language element* like *beer* (see Table 5.6), already a small collection can produce some reasonable top-ranked terms, yet the quality of the similarities quickly degrades when computing the DT on 100K and 1M sentences. A typical effect for the largest of our corpora is illustrated with *lager*, a sort of beer. Whereas this term is about 20 times less frequent than *beer*, it can collect enough significant contexts to enter the top 10 list. We frequently observe that rather rare hyponyms and co-hyponyms are similar to targeted *language elements* in the DTs computed from 105M sentences, which tremendously increases coverage for applications.

Looking at the most similar terms for the infrequent noun *ion* (see Table 5.7), we receive rather random collections of terms when using 100k and 1M sentences for the similarity computation. Whereas the averaged scores for the DTs using 100k and the 1M sentences are equal, the quality for the 100k-based DT appears to be lower. As described in Section 5.4.2, we ignore terms, which are not contained in WordNet. Thus, in the 100k-based DT, the four similar terms occurring within the top 10 ranked terms (*BB*, *LEAD*, *they*, *C*) are neglected during the evaluation, which is the reason for the equal scores. Again, using larger corpora results to higher similarity scores. We also observe a higher specialization of similarities for the largest corpus. Whereas the scores for the 10M- and the 105M-based DT are equal, the ranking is better for the DT computed with 105M sentences. This can be proofed by computing the *P*@5, where we obtain a score of 0.15 when using 10M sentences, and a score of 0.21 using 105M

Language element: **ion**

rank	100k freq:	WP 2	1M freq:	WP 23	10M freq:	WP 200	105M freq:	WP 2027
1	drought	1/13	battery	1/10	particle	1/2	particle	1/2
2	card	1/11	frame	1/6	electron	1/5	atom	1/10
3	production	1/8	car	1/10	oxygen	1/11	electron	1/5
4	winner	1/9	rocket	1/10	molecule	1/10	molecule	1/10
5	chain	1/8	tank	1/10	acid	1/13	proton	1/6
6	west	1/8	bomb	1/9	proton	1/6	chloride	1/13
7	frame	1/6	market	1/11	re	1/12	dioxide	1/14
8	digger	1/10	housing	1/8	gas	1/10	oxide	1/13
9	dust	1/11	recommendation	1/13	copper	1/13	gas	1/10
10	hydrogen	1/11	bottle	1/10	iron	1/9	acid	1/13
Ø		0.11		0.11		0.15		0.15

Table 5.7: This table presents DT entries, computed with a *Stanford holing operation*, for the infrequent noun *ion* with WordNet Path scores (WP), comparing different corpus sizes from 100K sentences up to 105M sentences

sentences for computing the DT. Additionally, we detect a coverage problem of WordNet. For example the term *atom* gets a very low path score, although an ion is an electrically charged atom and should have a much shorter path.

In the previous experiments, we weighted the *context features*, two terms share after the pruning strategies, equally. Thus, the parameter p becomes relevant, as we have to find a good setting for this parameter. Due to the pruning strategies, each term has at most p *context features*. Computing the cosine similarity between all *context features* would prevent the scalability of our approach. But we also have the parameter *wpfmax*, which filters features that co-occur with too many terms. During the aggregation of the *language elements*, the number of terms, a feature occurs with, is already known. Thus, we can use this number in order to define a relevance score for a *context feature*. We define the following three $scoring(x, y)$ functions as shown in Table 5.8.

name	**scoring**
one	scoring(x,y)= 1
inverse	scoring(x,y)= $1/\lvert X_{y,pruned}\rvert$
inverse-log	scoring(x,y)= $1/\log_2(\lvert X_{y,pruned}\rvert)$

Table 5.8: Scoring function to give weights to context features based on co-occurrence between *language elements* and *context features*.

The simplest scoring, used in the previous experiments, applies a weighting of one if a *language element* co-occurs with a *context feature*. We expect that *context features* occurring with too many *language elements* do not contribute much information, as in most cases these

context features are very general (e.g. stopwords). This is also the reason, why we remove *context features* that co-occur with more than 1000 *language elements* (*wpfmax* = 1000).We demonstrate this explicitly, by scoring the *language element-context feature* co-occurrence not by one but using a score, which rewards features that occur with few words more than features that occur with too many words. We implement this mechanism by adding the inverse of the number of terms a feature shares $|X_{y,pruned}|$, resulting to the *inverse scoring*. As the number of terms a feature occurs with is power-law distributed (see Section 2.6), we also compute the inverse of the logarithm of the number of terms per feature, which we call *inverse-log scoring*.

We demonstrate the performance for the different scoring methods using again the 10M sentences to compute DTs and consider the LMI significance measure. The results for frequent nouns using the one scoring, presented in Table 5.9, validate our intuition that using *context features* that occur with too many terms (*wpfmax* > 5000) does not yield the best results for computing similarities. Contrary, a strict pruning (*wpfmax* ≤ 100) results to lower scores. Comparing the one-scoring to the inverse scoring we observe consistently lower scores for the inverse ranking, except for the top 10 ranked words with *wpfmax* = 10000.

	scoring	**maximal terms per feature (*wpfmax*)**				
		10	100	1000	5000	10000
top 10	one	0.2361	0.2763	0.2847	0.2833	0.2802
top 10	inverse	0.2321	0.2706	0.2799	0.2817	0.2810
top 10	inverse-log	**0.2339**	**0.2761**	**0.2869**	**0.2883**	**0.2877**
top 5	one	0.2663	0.3067	0.3179	0.3187	0.3144
top 5	inverse	0.2592	0.3035	0.3156	0.3154	0.3143
top 5	inverse-log	**0.2625**	**0.3080**	**0.3219**	**0.3233**	**0.3240**

Table 5.9: WordNet Path Scores for DTs computed on 10M sentences. The DTs are computed with different settings of the *scoring* and the parameter *wpfmax* for the top 5 and top 10 terms considering frequent terms.

Using the logarithmic based scoring (inverse-log), results to the most stable results. We achieve the best results and as *context features* with many *language elements* have a low contribution, the scores do not decline when using high values for *wpfmax*. Similar trends are observed for the evaluation based on infrequent nouns as shown in Table 5.10.

The inverse scoring, results to lower scores compared to the one-scoring and the best results are obtained with the *log-scoring*. We advise to use the log-scoring when trying to achieve good results and having little knowledge about the data. In terms of simplicity, we use the *one-scoring* with $p = 1000$ and *wpfmax* = 1000 if not otherwise noted.

	scoring	maximal terms per feature (*wpfmax*)				
		10	100	1000	5000	10000
top 10	one	0.1600	0.2034	0.2133	0.2135	0.2106
top 10	inverse	0.1578	0.1909	0.2009	0.2012	0.2017
top 10	inverse-log	**0.1606**	**0.2041**	**0.2159**	**0.2183**	**0.2172**
top 5	one	0.1786	0.2259	0.2406	0.2392	0.2348
top 5	inverse	0.1766	0.2161	0.2262	0.2276	0.2269
top 5	inverse-log	**0.1798**	**0.2292**	**0.2437**	**0.2472**	**0.2465**

Table 5.10: This table shows results of the WordNet Path scores for DTs that are computed with different settings of the *score* and the parameter *wpfmax* for the top 5 and top 10 terms considering infrequent terms. The DTs are computed based on 10M sentenced.

5.7.2 The Impact of Different Holing Operations and Significance Measures

In this section, we examine the impact of different significance measures based on different holing operations. We use the Stanford collapsed dependency parse, the bigram and trigram holing operation (see Section 2.4). The latter two do not rely on any language specific preprocessing. For the computation we do not filter *language elements* and *context features* and use $wpfmax = 1000$ and $p = 1000$. The evaluation is again based on the WordNet Path similarity scores for the 5 top-ranked terms based on 1000 frequent and 1000 infrequent nouns.

First, we show results for the Stanford-based models both for frequent and infrequent nouns in Figure 5.3. In both graphs we plot the average WordNet Path score against the log-scaled corpus size. As already examined in Section 5.7.1, the PMI measure results to the lowest

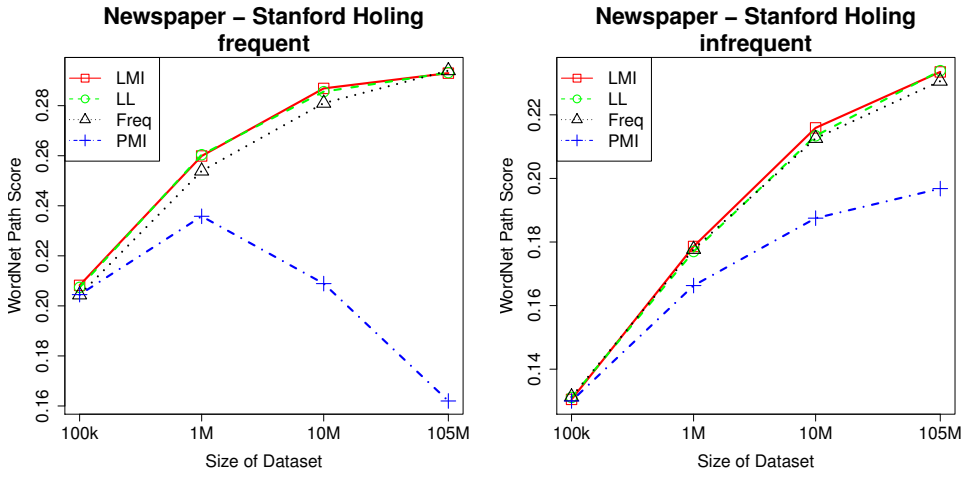


Figure 5.3: Comparing the corpus size in log-scale against different significance measures based on the Stanford holing operation for the frequent (left) and infrequent nouns (right).

scores and does not scale well to larger data according to the results for the frequent terms. This can be attributed to the fact that PMI favors *context features* that rarely co-occur with *language elements* (Bordag, 2008). It is surprising that recent works still keep using PMI for weighting *language elements* and *context features* in order to calculate similarities between *language elements* (Goyal et al., 2010; Pantel et al., 2009). However, these approaches evaluate their approach only with respect to their own implementation or extrinsically and do not prune on saliency. For infrequent *language elements*, where not that many *context features* are pruned, due to their lower frequency, the WordNet Path scores are much lower than for the other significance measurements. Examining the frequent nouns (left graph in Figure 5.3), we observe that the results for LL and LMI are alike for all different corpus sizes. The frequency based ranking achieves lower scores for the corpora below 105M sentences in comparison to LL and LMI. However, the frequency-base DT surmounts DTs computed with both measures when applied on the 105M corpus. For the evaluation based on the infrequent nouns (see right graph in Figure 5.3) we observe that the frequency is below the LL and is in line with the LMI except for the 105M sentences, where it is below the LL and LMI measures that achieve the highest scores. Whereas for infrequent nouns the PMI-based DTs score lower in comparison to DTs computed with other measures, we observe an increase of the WordNet Path score with increased corpus size.

In Figure 5.4 we present the performances for the significant measurements using the bigram holing operation. In comparison to the results examined using the Stanford holing operation, the average WordNet Path scores are consistently lower. Again, we observe that using larger corpora improves the performance. Thus, we could achieve a DT of high quality using a simple context representation if we have large amounts of text.

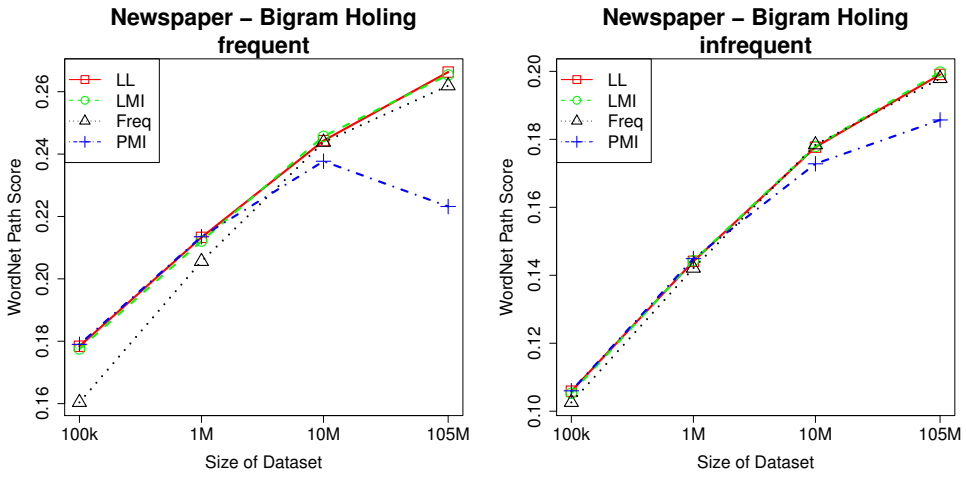


Figure 5.4: Comparing the corpus size in log-scale against different significance measures based on the bigram holing operation for the frequent (left) and infrequent nouns (right).

Interestingly, here the DTs computed with the PMI measure are en par with the LMI- and LL-based DTs both for frequent and for infrequent nouns when they are computed on 100k and 1M sentences. However, using more sentences, the scores achieved with the DTs, computed with the PMI measure, decline again and yield the lowest scores when computed on more than 10M sentences. The results for *Freq* are lower for the 105M sentences in comparison to the LL and LMI measure and are lowest for the 100k and 1M corpus for both the frequent and infrequent nouns. Again, using the LL and the LMI measures for ranking *context features* results to similar WordNet Path scores and achieves the highest scores for both frequent and infrequent nouns.

The third holing operation is the trigram holing operation, which uses both, the left- and the right-neighboring word at the same time to represent the context of a term. Results of this operation indicate a same trend as using the bigram holing operation, as can be observed in Figure 5.5.

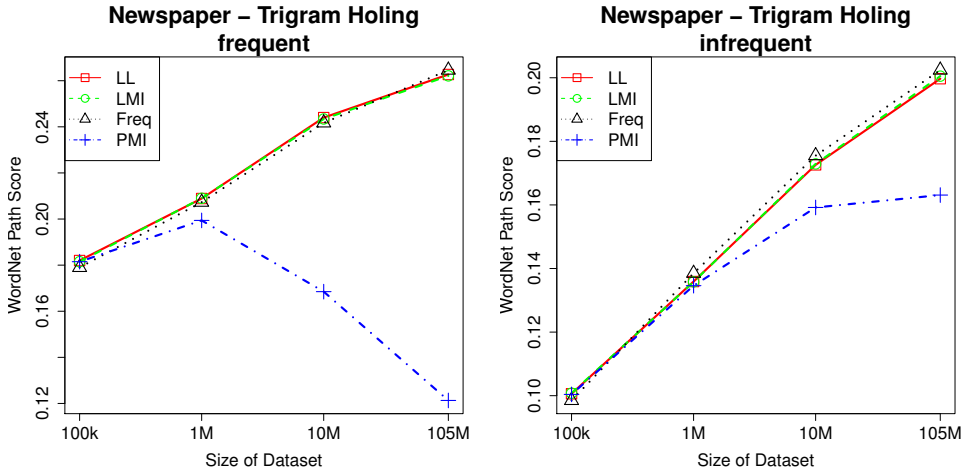


Figure 5.5: The two graphs show WordNet Path scores, illustrating the impact of the corpus size and different significance measures based on the trigram holing operation for the frequent (left) and infrequent nouns (right).

However, the curves of DTs computed with PMI are more similar to the ones using the Stanford dependency parse holing operation. Using the smallest sampled corpus, the choice of significance measure has only a marginal influence on the performances. However, already using 1M sentences, the PMI measure falls short of performance against the other ranking measures. Again, the PMI measure cannot retrieve relevant context features when computed based on large corpora for frequent nouns. Examining evaluation results for both frequent and infrequent nouns, the scores using LL, LMI and Freq are similar. In this experiment the frequency (Freq) achieves the highest scores for frequent nouns using 105M sentences and

outperforms the other measures for the infrequent nouns using more than 100k sentences for computing DTs.

Language element: **ion**

rank	Bigram	WP	Trigram	WP	Stanford	WP
1	lithium	1/12	diesel	1/10	particle	1/2
2	battery	1/10	electron	1/5	atom	1/10
3	treatment	1/14	search	1/12	electron	1/5
4	channels	1/15	cable	1/9	molecule	1/10
5	maker	1/10	combustion	1/13	proton	1/6
6	cell	1/7	jet	1/12	chloride	1/13
7	use	1/11	proton	1/6	dioxide	1/14
8	hybrid	1/8	in	1/12	oxide	1/13
9	amino	1/11	distribution	1/13	gas	1/10
10	silicon	1/11	laser	1/10	acid	1/13
Ø		0.0962		0.1082		0.1469

Table 5.11: The top 10 highest ranked terms from three DTs computed on 105M sentences, using different holing operations for the infrequent term *ion*. Additionally, we show the WordNet Path scores for each term.

For an insight into the resulting DTs using different holing operations, we exemplary present the top ten similarities for two terms extracted from DTs that are computed on the largest corpora using the LMI measure. In Table 5.11, we show the similar terms, which also occur in WordNet, for the infrequent term *ion* including the WordNet Path score.

According to the average WordNet Path score, the best results are achieved with the Stanford holing operation. Comparing the similarities observed with the two language-independent holing operations (bigram and trigram holing operation), the trigram holing operation achieves similarities that are closer to near-synonymy. For example we observe the terms *electron* and *proton* that are not found within the top ten entries using the bigram-based DT.

The similar entries for the frequent term *beer*, shown in Table 5.12, confirm this finding. We observe similar entries for both the trigram- and the Stanford-based holing operation, which are mostly beverages, and have an overlap of 5 terms. Using the bigram holing operation we get similarities to topically related terms for beer (e.g. *brewery* or *drinking*) rather than beverages. Comparing the bigram holing operation to the Stanford holing operation only 3 similar entries are shared. Inspecting how many terms are not contained in WordNet when finding the top ten similar terms we find nine for the bigram-, four for the trigram- and none for the Stanford-based holing operation. Most of these terms are plural word forms and brand names e.g. *beers*, *Anheuser-Busch* or *Gatorade*. These issues do not arise for the Stanford model, as words are lemmatized and use POS tags, which alleviates brand names, as they are not common but proper nouns. This might arise the question, whether the performance of the bigram- and trigram-based DTs would improve when using lemmas and POS tags. We performed such

Language element: **beer**

rank	Bigram	WP	Trigram	WP	Stanford	WP
1	brewery	1/15	wine	1/4	wine	1/4
2	liquor	1/4	coffee	1/5	drink	1/4
3	brewing	1/19	alcohol	1/3	coffee	1/5
4	drinking	1/13	soda	1/6	beverage	1/4
5	brewer	1/11	liquor	1/4	lager	1/2
6	drink	1/4	tea	1/5	soda	1/6
7	wine	1/4	champagne	1/6	tea	1/5
8	bottle	1/14	beverage	1/4	ale	1/2
9	alcohol	1/3	water	1/6	liquor	1/4
10	brew	1/2	lemonade	1/6	vodka	1/5
Ø		0.1942		0.2150		0.2767

Table 5.12: This table shows the top 10 highest ranked terms from three DTs using different holing operations for the frequent term *beer* including their WordNet Path scores. The DT have been computed based on 105M sentences.

an experiment based on the trigram holing operation in (Ruppert et al., 2015b). Whereas in that article the largest corpus consists of 10M sentences, the improvements using the trigram holing operation using lemmas and POS tags are marginal in comparison to the performance of an entirely language-independent preprocessing.

In addition to the similarities for the two *language elements*, we show the most significant *context features* that are shared between the terms *beer* and *wine* in Table 5.13.

For both the bigram and trigram holing operation, we observe different morphological variations of the same lexems. However, with these representations humans can infer that *beer* and *wine* are drinks, which are served in bottles and are goods that can be bought, sold and imported. Considering the collapsed dependencies from Stanford parser, similar relations are extracted which use a more precise context representation. The information that both *wine* and *beer* are alcoholic beverages is covered in all holing operations not only by the term *liquor* but also by the frequent occurring context noun *drinker*.

To summarize the influence of different significance measures and different holing operations, we show the evaluation results of DTs computed on 105M sentences in Table 5.14. We marked the best performing significance measurement for each holing operation with bold font. The highest WordNet path scores, computed for frequent nouns, are achieved for two holing operations when using the frequency (Freq) for ranking *context features*. Only for the bigram holing operation the highest WordNet Path scores are achieved in an evaluation with a DT that uses the LL measure. This is consistent for both the top 5 and the top 10 results. However, for infrequent nouns the best performance for the top 5 entries is gained for the LL-based DTs for two holing operations. Using the top 10 entries for computing the WordNet Path score, no distinct trend is observed. As already shown in Section 5.7.2, DTs computed

Context representations for *beer* and *wine*		
Bigram	Trigram	Stanford
(@;drinkers)	(a;@;bottle)	(dobj;drink#VB;@)
(@;bottles)	(drinking;@;.)	(prep_of;bottle#NN;@)
(@;bottle)	(of;@;bottles)	(nn;bottle#NN;@)
(drinking;@)	(drink;@;and)	(nn;drinker#NN;@)
(drink;@)	(drink;@;,)	(dobj;sip#VB;@)
(@;drinker)	(drinking;@;,)	(prep_of;glass#NN;@)
(@;tasting)	(empty;@;bottles)	(conj_and;@;liquor#NN)
(@;consumption)	(cheap;@;and)	(conj_and;wine#NN;@)
(drank;@)	(American;@;drinkers)	(conj_and;@;wine#NN)
(bottled;@)	(and;@;bottles)	(nn;consumption#NN;@)
(@;lovers)	(of;@;drinkers)	(conj_and;@;liquor#NN)
(@;glasses)	(of;@;sold)	(dobj;@;pour#VB)
(imported;@)	(drinking;@;and)	(nn;tasting;@;NN)
(sipping;@)	(a;@;drinker)	(prep_of;@;sip#NN)
(premium;@)	(the;@;bottles)	(nn;@;selection#NN)
(@;distributor)	(buying;@;,)	(nn;drinking#NN;@)
(@;shipments)	(,;@;consumption)	(nn;@;distributor#NN)
(@;coolers)	(a;@;salesman)	(nn;@;drinking#NN)
(sipped;@)	(serve;@;and)	(nn;@;lover#NN)
(@;cooler)	(broken;@;bottle)	(nn;@;cooler#NN)
(@;geeks)	(drank;@;in)	(conj_and;spirit#NN;@)

Table 5.13: This table shows the 20 highest ranked context features that the terms *beer* and *wine* share. The context features are extracted from DTs computed on 105M sentences that are computed for three different holing operations.

with the PMI measure score lowest in the evaluation. The second best result, which is still close to the highest scores, is achieved in most cases with DTs using the LMI measure. As the LL measure is complex to compute, we advise to use either the LMI measure or the frequency for all holing operations. The surprisingly good performance for the usage of the frequency is in accordance to the findings by Padró et al. (2014).

Comparing the three holing operations, we obtained best results when using the collapsed dependency parse holing operations even when using smaller corpora. With the increase of the corpus, still using the dependency parse holing operation is advised and yields the best performance. However, the distance between the scores decreases. This is consistent with a previous, similar evaluation performed by Curran (2004). We also notice higher WordNet Path scores when using the bigram holing operation for computing similarities. Inspecting the most similar terms (see Tables 5.11 and 5.12), the similarities, computed with the usage of the bigram holing operation, seem to be topically related. The similarities obtained with the trigram operation seem to be more near-synonymy related and are closer to the ones achieved with the Stanford dependency parser holing operation. As computing dependency

	top 5				top 10			
	LMI	LL	Freq	PMI	LMI	LL	Freq	PMI
frequent nouns								
Bigram	0.2940	**0.2943**	0.2931	0.2514	0.2654	**0.2663**	0.2619	0.2232
Trigram	0.2914	0.2902	**0.2950**	0.1549	0.2621	0.2627	**0.2643**	0.1213
Stanford	0.3301	0.3301	**0.3345**	0.1700	0.2930	0.2932	**0.2936**	0.1620
infrequent nouns								
Bigram	0.2241	**0.2254**	0.2220	0.2090	**0.1998**	0.1991	0.1979	0.1857
Trigram	0.2277	0.2290	**0.2297**	0.1875	0.2003	0.1997	**0.2024**	0.1631
Stanford	0.2663	**0.2669**	0.2623	0.2227	0.2337	**0.2339**	0.2303	0.1968

Table 5.14: This table shows results for the WordNet Path evaluation scores for DTs computed based on 105M newspaper sentences. We present the performance of different holing operations using different significance measures on frequent and infrequent nouns. For each holing operation we highlight the best performing significance measure in bold font.

parses is time-consuming, we would advise to use the trigram holing operation when no parser is available or parsing is too time-consuming, which is the case, especially, for large corpora.

5.7.3 Different Evaluation Methods

In the previous sections, we have used the WordNet Path measure to evaluate our framework and its parameters. Whereas Lin (1998) has already used WordNet for the DT evaluation, we were the first, who used the WordNet Path measure for evaluating DTs (Biemann and Riedl, 2013). However, applying a manually created thesaurus for the evaluation, as done by Curran (2004), seems to be more convenient. Thus, we apply the thesaurus-based evaluation as described in Section 5.4.1 and compare it against the WordNet Path measure evaluation. For this, we repeat the experiment using the Stanford collapsed dependency holing operation considering several significance measures for ranking the context feature, which was presented in Section 5.7.1 in Figure 5.3. We consider three measures for comparing a manually created thesaurus to an automatically generated thesaurus: P@1, Inverse Ranking and GAP (see Section 5.4). Whereas the P@1 only considers the first similar entry in the DT, the GAP and the inverse ranking measure consider the top 200 entries. As the inverse ranking measure does not seem to be a standard measure for ranking, we use the GAP, which is a common evaluation measure for ranking (e.g. Szarvas et al. (2013), Thater et al. (2009)) and results to scores between 0 and 1, with 1 indicating a perfect ranking.

We show the results on the frequent nouns on the top graph in Figure 5.6. The graphs on the bottom present the evaluation results for infrequent nouns. From left to right we plot results based on the P@1, the inverse ranking and the GAP measure.

For all evaluation metrics, we observe different ranges of scores. In addition to the scores, the graphs for all three metrics based on the thesaurus measure are highly correlated both

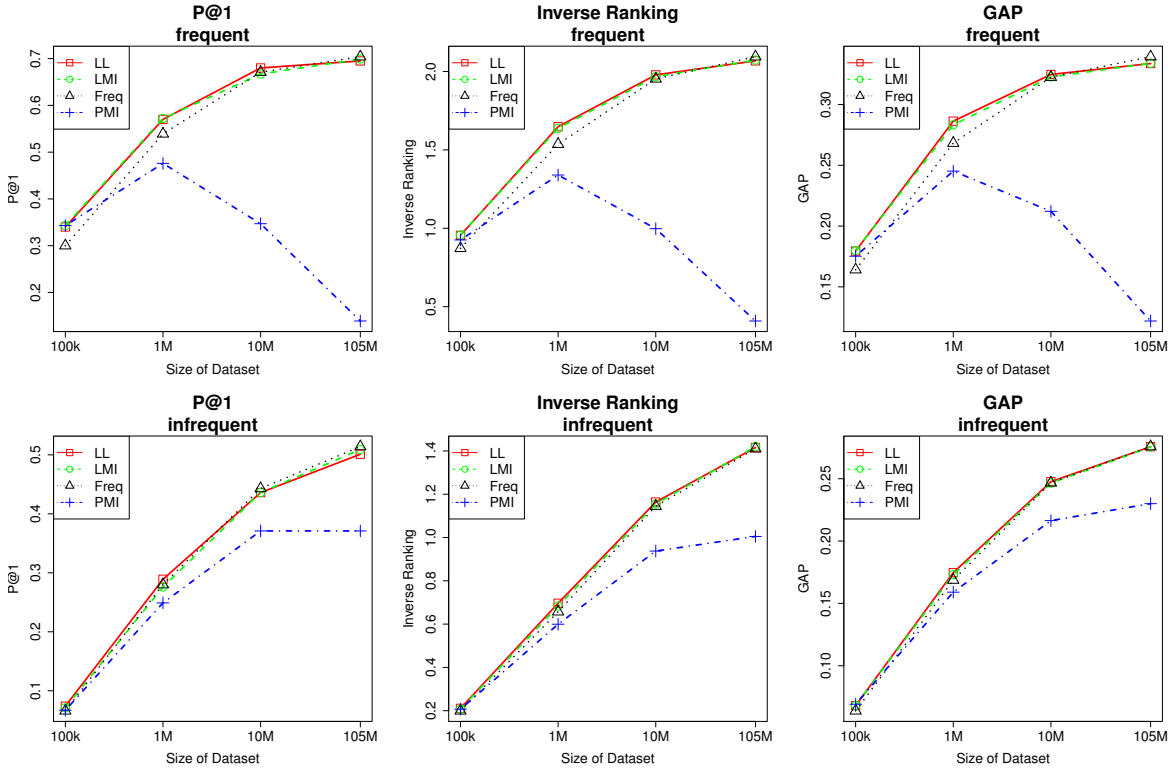


Figure 5.6: Results of three different evaluation measures based on manually created thesauri. We show the results for the 1000 frequent nouns on the top and for infrequent nouns on the bottom of this figure.

for frequent and infrequent nouns. The only major difference is observed for the infrequent nouns for the LL-based DT using 105M sentences. When computing the P@1, we observe the highest scores using the LL-based DT. Considering the GAP and the inverse ranking measure for the evaluation, we notice lower scores than using Freq- and LMI-based DTs. As the P@1 only considers the most similar entry for each noun, the measure is more prone for spurious rankings. Nevertheless, we only observe a marginal difference. Comparing the results achieved with the thesaurus based evaluations with the results of the WordNet Path measure shown in Figure 5.3, we also observe a high correlation.

The findings of this section validate the usage of the WordNet Path measure. The evaluation against a manually created thesaurus appears to be intuitively better suited than using WordNet. However, we observe a high correlation when using a different resource and three different evaluation measures to evaluate our DT. Using WordNet for the evaluation can be performed automatically and thus reduces efforts to collect data, as easy access to WordNet is available. Building a gold standard thesaurus following Curran and Moens (2002) requires access to several thesauri. Whereas for some, programming interfaces exist, often with limited

access and license restrictions, others have to be extracted manually. Thus, we continue using the WordNet Path measure for the remaining experiments.

5.7.4 Comparison to Other Similarity Computations

In this section, our similarity computations are compared against three measures for computing DTs. First, we use the best performing measures from the symbolic approaches introduced by Lin (1998) and Curran (2002). Furthermore, we compare our approach to word embeddings (Mikolov et al., 2013). All methods are computed based on the newspaper corpus, which consists of 105 million sentences (see Section 5.6).

Curran's measure does not rely on any specific *context feature* representation and thus, we can directly convert it using our tuple-based representation. Lin's measure assumes dependency relations as contextual representation. Additionally, Lin's measure requires the frequencies for the dependency relation r. For computing their approaches, we use the Stanford collapsed dependency parse holing operation. For Lin's measure, we split the *context feature* from y into the elements (r, y_t) with y_t denoting the dependent term and r the relation name. Comparing both approaches formula-wise, they specify the similarity of terms using an "information measurement" for each term-context tuple and then calculate the similarity between terms using similarity measures. We show our measure and the measures used by Lin (1998) and the measure recommended by Curran (2002) in Table 5.15.

Comparing our approach to other distributional similarity measurements (cf. Lee, 1999; Lin, 1998; Weeds, 2003), we consider a simpler information measure, which is always 1, as shown in Table 5.15.[43] Our scoring function is not pre-computed and thus, we can compute similarities directly after the pruning step and avoid a "two-staged" computation. Aside from our information measure, using only the p features per *language elements*, which have the highest significance scores, speeds up the runtime tremendously and acts as a noise filter as unimportant *context features* are excluded. Furthermore, our similarity measure is computationally simpler as we only use the sum of shared features, which are significant and do not require a specific similarity measurement.

In Figure 5.7, we present the WordNet Path scores for Curran's, Lin's and our method, which have been computed on different corpus sizes. The left graph shows the evaluation results for frequent nouns and the right graph for infrequent nouns. Based on the frequent nouns, Curran's similarities yield the lowest path scores. We could not confirm that his measure outperforms Lin's measure as stated in (Curran, 2002), which is in line with the results presented by Kiela and Clark (2014).[44] An explanation for his different evaluation results might be the usage of a different parser, the small amount of test words and a different gold stan-

[43] A list of different information measures used in our framework are listed in Table 5.8 in Section 5.7.2.

[44] Regarding Curran's Dice formula, it is not clear whether to use the intersection or the union of the features. We use an intersection, as it is unclear how to interpret the minimum function otherwise, and the alternatives performed worse.

Information Measurements	
Lin's formula	$I(x,y) = lin(x,y) = \log \frac{\lvert\langle x,y\rangle\rvert \cdot \lvert\langle *,(r;*)\rangle\rvert \lvert\langle *,y\rangle}{\sum(\lvert\langle x,(r;*)\rangle\rvert)\lvert\langle x,*\rangle\rvert}$
Curran's t-test	$I(x,y) = ttest(x,y) = \frac{\frac{\lvert\langle x,y\rangle\rvert}{\lvert\langle X,Y\rangle\rvert} - \frac{\lvert\langle x,*\rangle\rvert}{\lvert\langle X,*\rangle\rvert} \cdot \frac{\lvert\langle *,y\rangle\rvert}{\lvert\langle *,Y\rangle\rvert}}{\sqrt{\frac{\lvert\langle x,*\rangle\rvert}{\lvert\langle X,*\rangle\rvert} \cdot \frac{\lvert\langle *,y\rangle\rvert}{\lvert\langle *,Y\rangle\rvert}}}$
Our scoring	$\text{scoring}(x,y) = 1$
Similarity Measurements	
Lin's formula	$sim(x_i,x_j) = \frac{\sum_y^{Y_{x_i} \cap Y_{x_j}} (I(x_i,y)+I(x_j,y))}{\sum_y^{Y_{x_i}} I(x_i,y) + \sum_y^{Y_{x_j}} I(x_j,y)}$
Curran's dice	$sim(x_i,x_j) = \frac{\sum_y^{Y_{x_i} \cap Y_{x_j}} min(I(x_i,y),I(x_j,y))}{\sum_y^{Y_{x_i} \cap Y_{x_j}} (I(x_i,y)+I(x_j,y))}$
measure in this work	$sim(x_i,x_j) = \sum_y^{Y_{x_i,pruned} \cap Y_{x_j,pruned}} scoring(x_i,y) = \sum_y^{Y_{x_i,pruned} \cap Y_{x_j,pruned}} scoring(x_j,y)$

Table 5.15: Similarity measures used for calculating the distributional similarity between terms.

dard thesaurus. Comparing our method, which is computed using LMI, to Lin's method, we achieve lower scores with our method using small corpora, but surpass Lin's measure beyond 10 million sentences. This is in line with results, we presented in Riedl and Biemann (2013a).

Considering the infrequent nouns (right graph in Figure 5.7), we consistently receive the highest scores with our LMI method. Again, we observe that our method improves over Lin's method especially when using larger corpora. Once more, the lowest scores are obtained using Curran's method.

In addition, we compare our method against word embeddings (Mikolov et al., 2013), using the `word2vec` implementation (see Section 3.2.4), which have become popular in recent years. The standard method, is computed based on neighboring words and does not support the usage of syntactic dependencies. However, a more general approach has been proposed by Levy and Goldberg (2014b), which allows the computation of word embeddings using syntactic dependencies.

First, we compare the standard word embeddings with DTs that have been computed based on neighboring words using the bigram and trigram holing operation from the previous section. For the computation of the word embeddings we use `word2vec` and consider two different parameter settings. First we compute the models using the program's default parameters.[45]

[45]The default parameters from `word2vec` are as follows: `-size 100 -window 5 -sample 1e-3 -negative 5 -hs 0 -iter 5`.

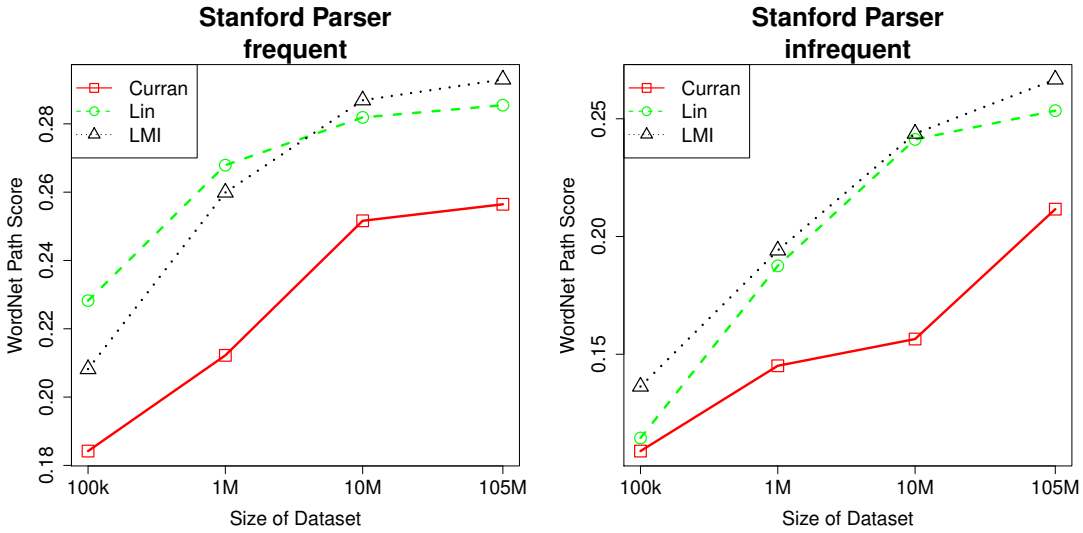


Figure 5.7: Comparing the corpus size in log-scale against the WordNet Path score for the top 10 words extracted from our method (LMI), Lin's and Curran's semantic models that are computed on Stanford dependency parses. We show results for frequent (left) and infrequent nouns (right).

Additionally, we use "advanced parameters", which are used in the example shell script of `word2vec`.[46] In contrast to the default parameters, a higher number of dimensions is used, more iterations are considered for the training and the number of negative examples is higher.

In contrast to our context representation, `word2vec` uses further hyperparameter e.g. negative examples and larger context windows. In order to get an impression about the quality of the ranking, we do not only show results for the 5 and 10 most similar entries from the models but also for the top 20, 50 and 100 most similar terms.

We observe that both word embedding models, computed with the default parameter of `word2vec`, score consistently lower than our models for frequent nouns based on the WordNet Path evaluation. For infrequent nouns, the default CBOW and Skip-gram embeddings score slightly higher than our bigram- and trigram-based models but only when considering the top 5 most similar terms. In an evaluation using the top 10 most similar terms only the CBOW model yields higher scores. Considering more than the top 20 most similar entries our methods achieve consistently higher scores against both word embedding models.

Using advanced parameters for computing the word embedding models with `word2vec`, we observe higher scores for the similarities retrieved from the CBOW model than for the similarities extracted from our bigram and trigram holing operation for the top 5 for both frequent and infrequent words. Performing the evaluation using the top 10 most similar terms, the advanced CBOW model results to comparable performances compared to our bigram- and

[46] We use the parameters suggested in the script `demo-word.sh` as advanced parameters. These are defined as follows: `-size 200 -window 5 -sample 1e-4 -negative 25 -hs 0 -iter 15`.

	Our Methods			Word Embeddings			
	Stanford Parser	Trigram	Bigram	CBOW default[45]	CBOW advanced[46]	Skip-gram default[45]	Skip-gram advanced[46]
	frequent nouns						
top 5	0.3301	0.2914	0.2940	0.2836	0.3045	0.2646	0.2922
top 10	0.2933	0.2621	0.2654	0.2523	0.2656	0.2184	0.2548
top 20	0.2610	0.2340	0.2399	0.2261	0.2331	0.1685	0.2108
top 50	0.2250	0.2023	0.2075	0.1918	0.1919	0.1009	0.1366
top 100	0.2019	0.1800	0.1858	0.1553	0.1453	0.0578	0.0789
	infrequent nouns						
top 5	0.2663	0.2277	0.2241	0.2282	0.2461	0.2285	0.2413
top 10	0.2337	0.2003	0.1998	0.2005	0.2105	0.1939	0.2068
top 20	0.2054	0.1786	0.1784	0.1773	0.1836	0.1624	0.1744
top 50	0.1778	0.1562	0.1578	0.1498	0.1513	0.1142	0.1259
top 100	0.1611	0.1391	0.1422	0.1230	0.1139	0.0737	0.0788

Table 5.16: Similarities computed using 105M sentences of newspaper data. We show WordNet Path scores using our methods with the Stanford Parser, the trigram and the bigram holing operation. Additionally, we show scores based on embeddings using CBOW and Skip-grams computed with `word2vec`.

trigram-based models for frequent nouns. However, the similarities from the Skip-gram word embeddings, computed with advanced parameters, yield lower scores in comparison to the similarities from our bigram method in the evaluation using frequent nouns. By extending the evaluation to the highest 100 most similar words we observe that using the word embeddings, results to inferior scores in comparison to the scores obtained with our methods. While examining the similarities for the embeddings using frequent nouns, we detect high scores for the similarities in the top 5 entries and see a consistent decline of the similarity scores. Data analysis reveals that word embedding models prefer to rank infrequent terms higher. This is in particular observed for the similarities computed with the Skip-gram model, which yields low scores for the top 100 nouns. A similar trend is observed for the infrequent nouns. For the top 5 words, the word embeddings, computed with advanced parameters, outperform our method. But considering the top 50 words and beyond our bigram- and trigram-based similarities achieve higher WordNet Path scores.

This highlights, that the parameter tuning is essential for word embeddings. We have shown that computing word embedding models with the default parameter result to performances, which are inferior to the results with our method for frequent nouns and comparable to our results for infrequent nouns. Tuning the parameters of `word2vec` by using a higher dimensional vector representation and increasing the number of negative examples yields to higher runtimes but also improves the performance as demonstrated by higher WordNet Path scores. However, word embeddings seem to rank infrequent term higher than frequent ones. Thus, using our approach is advised when lower ranked similar words are required. However,

using syntactic dependencies in our method still outperforms the results of word embeddings computed with the advanced parameters.

Thus, we additionally compare our method against word embeddings computed on syntactic dependency parses. For this, we use an adaptation of `word2vec` called `word2vecf` proposed by Levy and Goldberg (2014b)[47]. For the computation of these embeddings we apply the parameters recommended by Levy and Goldberg (2014b)[48] and show the results in Table 5.17.

	Our Method	**Dependency-based Embeddings**	
	Stanford Parser	Skip-gram > 100	Skip-gram > 100 & dep. filter
	frequent nouns		
top 5	0.3301	0.3066	0.3054
top 10	0.2933	0.2630	0.2632
top 20	0.2610	0.2275	0.2277
top 50	0.2250	0.1884	0.1887
top 100	0.2019	0.1548	0.1549
	infrequent nouns		
top 5	0.2663	0.2249	0.2258
top 10	0.2337	0.1981	0.1992
top 20	0.2054	0.1764	0.1769
top 50	0.1778	0.1544	0.1543
top 100	0.1611	0.1374	0.1376

Table 5.17: Similarities computed using 105M sentences of newspaper data. We show WordNet Path scores using our methods with the Stanford Parser and show scores based on syntactic embeddings Levy and Goldberg (2014b) computed on the same corpus. In addition we show results with different filterings.

Comparing the dependency-based word embeddings, we mostly observe slightly higher results, when filtering syntactic dependencies. Furthermore, the dependency-based word embeddings score slightly higher than the advanced CBOW word embeddings (see Table 5.16) for the top 5 most similar term. However, in the evaluations considering more than the 5 most similar terms, they achieve lower scores than the advanced CBOW word embeddings. Computing our DT with syntactic dependencies, we achieve much higher scores in comparison to all results based on word embeddings, either using neighboring words or syntactic dependencies. A data analysis reveals that the dependency-based embeddings suffer even more, than the word-based embeddings from infrequent terms, which are ranked at high positions and obtain low WordNet Path scores.

[47] An implementation is available here: `https://bitbucket.org/yoavgo/word2vecf`

[48] We remove words and context occurring less than 100 times in the corpus and use the following parameters: `-size 300 -negative 15`.

In summary, the results shown in this section validate our pruning approach. Whereas Lin and Curran propose measures to weight term-feature co-occurrences, they do not remove features that occur with too many terms, which is performed with our method. Using the pruning steps of our method enables the usage of a less complex similarity measure. This reduces the computation times and additionally results to higher WordNet Path scores when using large corpora for the DT computation.

Comparing our bigram- and trigram-based DTs to word embeddings, computed with the default parameters, our models score higher for frequent nouns and comparable for infrequent nouns. Considering more advanced parameters we observe higher score for word embeddings in comparison to our bigram- and trigram-based models. Especially for the top 5 most similar terms for infrequent nouns, word embeddings outperforms our method. However, we observe a performance drop for the word embeddings when considering the top 50 most similar words and beyond. For lower ranked terms, our symbolic approach is able to detect more meaningful contexts in comparison to the numeric vector representations used by word embeddings and yields higher WordNet Path scores. Using syntactic dependencies as context for computing similarities, our approach achieves much higher WordNet Path scores than the word embeddings.

5.7.5 The Influence of Corpora

A further influence factor is the choice of corpus used to compute a DT. Using a corpus of a specific domain for computing a DT, will result to a domain specific vocabulary and to similarities targeted to this domain. Not only the size of the corpus is important for the quality of a DT, as shown in the previous section, but also the text quality. We demonstrate the influence of different corpora compared with their size. As first corpus we compute DTs on the previously introduced newspaper corpus of 105M sentences. The second corpus is a dump of the English Wikipedia (35M sentences) and as third corpus, we use the BNC corpus (5M sentences), which is probably the corpus of the highest quality. It was manually assembled with the goal to build a balanced corpus containing different texts representing the English language.[49] As largest corpus, we also apply Google Books[50] by using the syntactic n-grams from Google books (Goldberg and Orwant, 2013)[51] (17.6B sentences). For building the DTs for the newspaper corpus, the Wikipedia and the BNC we use the Stanford dependency parse holing operation with lemmatization.

We evaluate DTs that have been computed on downsampled pieces of the corpora. As this is only possible for corpora in textual form, we generated pieces of 100k, 1M and 10M sentences for the newspaper corpus, the Wikipedia corpus and the BNC. For the syntactic n-

[49] More details for these three corpora can be found in Section 5.6.

[50] `https://books.google.com/`

[51] The syntactic n-grams from Google books are available here: `http://commondatastorage.googleapis.com/books/syntactic-ngrams/index.html`.

grams extracted from Google books, we only consider the full n-grams as downsampling could be only performed per years. The evaluation is performed using the WordNet Path measure using the 10 most similar entries. We plot the scores against the corpus size (in log-scale) for the DTs computed on various corpora in Figure 5.8. The left graph shows results for frequent nouns and the right the results for infrequent nouns.

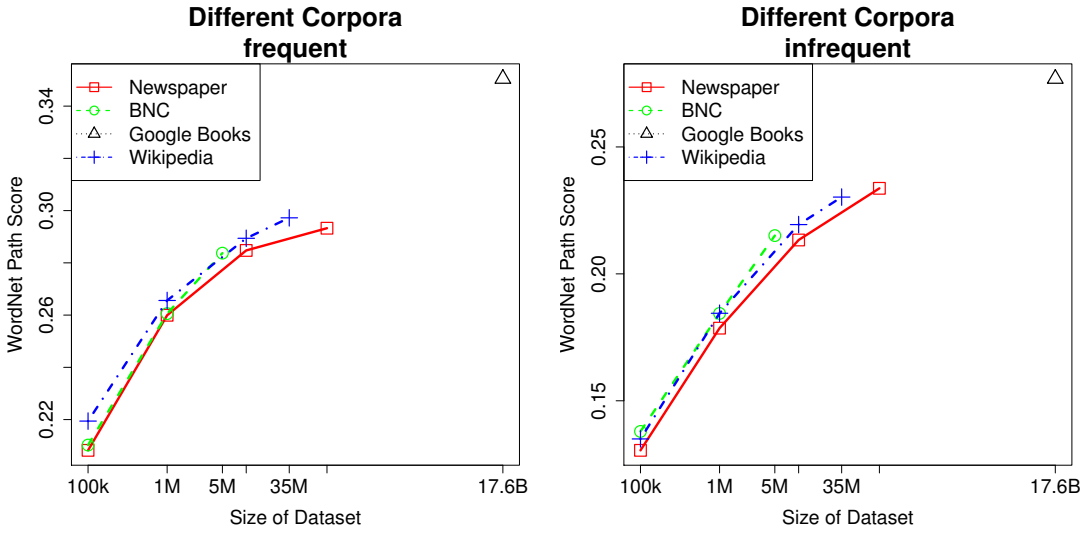


Figure 5.8: Comparison of the performance considering the top 10 entries from a DT against WordNet. Here we focus on the impact of different corpora. In addition we show results of the DTs when using downsampled corpora plotted. The corpus size is plotted in log-scale. The left graph presents the result for frequent nouns and the right graph shows results for infrequent nouns.

For frequent nouns, we obtain similar scores for DTs computed on the newspaper corpus and BNC up to a corpus size of 1M sentences. The highest scores are received with a DT generated on Wikipedia. Using the entire BNC corpus for computing the DT, we get WordNet Path scores that are similar to the ones gained with the Wikipedia-based DT using the same sized corpus. In our evaluation the newspaper-based DTs consistently yield the lowest scores. A different trend is observed for the infrequent nouns. Here, the BNC-based DT performs on a par with the Wikipedia-based DT until 1M sentences and outperforms the Wikipedia-based DT when using 5M sentences. Again the lowest performance is obtained using the newspaper-based DT. However, the highest WordNet Path scores, both for the infrequent and frequent nouns, are observed for a DT that is computed on the Google books corpus. This confirms again that using more data improves the resulting DT. Additionally, this proves the scalability of our framework to large amounts of data.

In addition to the WordNet Path measure used for the evaluation, we also present results using the thesaurus-based evaluation (see Section 5.4.1) in Table 5.18. We highlight the best performing DTs and report results both for frequent and for infrequent nouns.

	Corpus	# sentences	WordNet-based		Thesaurus-based			
			Path@5	Path@10	P@1	P@5	Inv. Rank	GAP
freq. nouns	BNC	5M	0.3199	0.2837	0.6770	0.5018	1.9804	0.3221
	Wikipedia	35M	0.3395	0.2972	0.6840	0.5280	2.0796	0.3358
	Newspaper	105M	0.3301	0.2933	0.6990	0.5190	2.0718	0.3336
	Twitter	890M	0.2368	0.2161	0.2830	0.2466	0.9662	0.1938
	Google Books	16.7B	**0.3956**	**0.3505**	**0.7540**	**0.5830**	**2.3730**	**0.3780**
	Google Web1T	95B	0.2903	0.2543	0.3490	0.3624	1.3370	0.2473
infreq. nouns	BNC	5M	0.2398	0.2151	0.4340	0.2924	1.1126	0.2380
	Wikipedia	35M	0.2618	0.2302	0.5250	0.3602	1.3867	0.2718
	Newspaper	105M	0.2663	0.2337	0.5100	0.3726	1.4208	0.2753
	Twitter	890M	0.1848	0.1666	0.2470	0.1596	0.6326	0.1621
	Google Books	17.6B	**0.3246**	**0.2769**	**0.6460**	**0.4946**	**1.8909**	**0.3406**
	Google Web 1T	95B	0.2263	0.1981	0.3190	0.2524	0.9436	0.2165

Table 5.18: Evaluation of DTs computed on different corpora of various quality. We show results for frequent and infrequent nouns using different evaluation metrics.

Additionally, we present results for a DT that is computed on the Google's Web1T corpus[52]. This corpus contains n-grams from 95 billion sentences from various Web pages. As this corpus is available in form of n-grams, we cannot generate dependency parses of this corpus. Thus, we operate on the trigrams of this dataset and use the trigram holing operation for generating the contextual representation. Additionally, the Google Web1T-based DT does not contain POS tags and lemmatized terms. Whereas text from Web pages contains a lot of noise, tweets from Twitter[53] are "notoriously noisy" (Han and Baldwin, 2011). To show the performance of evaluating a Twitter-based DT against WordNet we crawled 1% of the daily Twitter data from 2012 to 2013 and filter out all non-English tweets. This corpus consists of 890M Tweets and is used for computing a DT with the trigram holing operation and without using further preprocessing. In Table 5.18 we ordered the results according to their corpus size. Again, the best results are achieved with the Google Books dataset for all evaluation metrics. This confirms again that size matters for the computation of DTs.

Additionally, the quality of text used for the computation is important. We observe a similar performance for the DTs computed on Wikipedia and newspaper. However, the newspaper corpus is three times larger than the Wikipedia corpus. The effect of using a very large corpus of lower quality and avoiding language-dependent preprocessing is shown for the results based on the Google Web1T-based DT. Whereas it is the largest corpus used in the evaluation, it achieves low scores. This is mainly attributed to the evaluation setting, as both WordNet and the manually used thesauri are targeted to the open text domain. Web data is biased to different domains, especially to domains that most often occur in Web data like the *porn* and *social*

[52] https://catalog.ldc.upenn.edu/LDC2006T13

[53] https://twitter.com/

network domains. To give examples, the term *guest* is most similar to the terms *DLR*, *dublin*, *Rating*, *matches* and *Mozilla*. These terms are related to the guest as a visitor of a website. As another example, the term *throat* is most similar to the terms *cumshot*, *bukkake* and *suck*, which can be assigned to the *porn* domain. These different meanings are not covered in WordNet and thus the Web 1T DT achieves low scores for some terms, which downgrade the overall performance using our evaluation. The lowest scores are observed using the Twitter-based DT. This can be attributed due to the low textual quality of tweets as well a completely different vocabulary in comparison to the vocabulary contained in a dictionary like WordNet. For example the word *memory* is most similar to the terms *sd* and *SD*, the abbreviation of *Secure Digital Memory Card*. In WordNet the term *SD* is an abbreviation of *South Dakota*. Additionally, the term *memory* is highly similar to the terms *debit* and *credit*, which are contained in WordNet but have a long WordNet Path distance. These terms are similar to *memory* as they are often used in conjunction with the term *card*. As a tweet is limited to 140 characters, words are often abbreviated, e.g. *b4* is used for *before*. Furthermore, tweets contain more spelling errors in comparison to Web and newspaper data.

Summarizing the results we achieve the best results leading to a P@1 of 0.7540 for the Google Books DT and 0.6990 using the newspaper corpus-based DT for frequent nouns and 0.6460 and 0.5100 for infrequent nouns. We show that using manually selected texts yield to a DT of higher quality. However, using large amounts of data for the DT generation is more important and achieves the highest scores. Additionally, we show that computing DTs using huge amounts of noisy text results to lower scores than performing the computation on smaller but cleaner corpora.

Whereas Riedl and Biemann (2013a) reported a runtime of less than 30 hours for the DT computation of the Google Books corpus, using a larger cluster (24 nodes with 256 cores) all similarity computations can be performed in less than 5 hours. Still, the most computation time is required for the language-dependent preprocessing like dependency parsing, POS tagging and lemmatization took almost a week on our cluster for the 105 million newspaper sentences. Computing a DT using huge corpora like Web1T or the Google Books Syntactic N-Grams would be intractable with standard vector-based measurements. For example Lin's and Curran's similarity measure could not be computable for the entire vocabulary of the newspaper corpus (see Section 5.7.4) with our Hadoop cluster, as too much memory would have been required. Thus, we computed similarities only for the 2000 test nouns on a server with 96GB of main memory.

5.7.6 Verb-based Evaluation

In the previous sections, we evaluated DTs solely on nouns. In this section, we switch the evaluation to verbs and follow the experimental setting used by Padró et al. (2014). They compute DTs following Lin (1998) based on the BNC. For the context extraction, they use

syntactic dependencies generated with the RASP parser (Briscoe et al., 2006). The evaluation is performed using the WordNet Path measure described in Section 5.4.2. They report the performance based on 3954 verbs extracted from BNC. Additionally, they show results for low (1509), mid (1227) and high frequent (1218) verbs. Using the same corpus and the same evaluation measure allows comparing our results to the ones reported by Padró et al. (2014). In contrast to Padró et al. (2014), we use the Stanford dependency parser for extracting the contextual representation and use the standard parameters for computing the DT as shown in Table 5.1. We do not use the RASP parser as Stanford support collapsed dependencies have shown to achieve better results in applications (e.g. Filippova and Strube (2008)). Furthermore, due to the statistic nature of the Stanford parser it is not restricted to work for English only.

We present the evaluation results for low, mid, high frequent verbs and for all verbs in Table 5.19. In accordance to Padró et al. (2014), we also perform the evaluation when computing DTs with different significance measures. Comparing the results, our method obtains higher scores

	Our method				Padró et al. (2014)
	low	mid	high	all	all
Freq	0.1879	**0.2336**	0.3208	**0.2430**	0.158
LMI	0.1863	0.2326	**0.3217**	0.2424	0.155
PMI	0.1692	0.2035	0.2773	0.2132	0.139
LL	0.1827	0.2261	0.3001	0.2323	-
Best by Padró et al. (2014)	**0.202**	0.154	0.208	0.164	

Table 5.19: WordNet Path measures for the top 10 entries for verbs split in different frequency groups. Additionally, to our results we also show the performance as reported by Padró et al. (2014) for all verbs as well as the best results achieved.

for frequent terms rather than for infrequent ones, which is the same trend as observed for nouns in Section 5.7.1. This is not the case for the results reported by Padró et al., which might be attributed to parameter optimizations for each result. Similar to the results by Padró et al., we achieve the best results with our method using the frequency (Freq) as weighting measure between *language elements* and *context features*, except for high frequent verbs. Our DTs obtain the highest scores, except for low frequent terms, without any further optimization. This might be due to different similarity calculations and pruning strategies. Whereas Padró et al. also apply the parameter p and wf, they do not remove *context features*, which co-occur with too many terms ($wpfmax$). As context features for verbs, Padró et al. select only syntactic dependencies to nouns including the relation name. Furthermore, they also apply a different dependency parser. Whereas their approach sweeps over different parameters for different frequency bands, our method uses the same parameters for all frequency bands. This might be a reason for the lower scores of our method for the low frequent verbs. Nevertheless, comparing the performance for all verbs (see the fifth column for our results and the sixth

column for the best performance reported by Padró et al. in Table 5.19) our methods yield much higher scores.

5.8 Conclusion

In this chapter, we have introduced a graph-based approach to compute distributional similarities between terms. We follow a symbolic representation that is able to process on various language elements and context representations in order to compute similarities. Using efficient pruning strategies and Hadoop's MapReduce paradigm, our approach scales to arbitrarily large data. This was demonstrated by computing a DT based on the syntactic Google *n*-gram dataset and Google's Web1T trigrams, which took less than five hours.

For the evaluation of our framework, we consider a WordNet-based distance measure and additionally use a direct comparison against manually created thesauri. Using the WordNet-based evaluation, we show the performance of using different parameters to provide the best working configuration set. As our framework does not rely on any specific *language elements* and *context features* used to compute similarities, we compared the performance of language-dependent context representations as well as language-independent ones. Whereas using syntactic dependency parsing to compute similarities consistently results to a DT, which scores highest in our evaluations, we observe reasonable performances when only using neighboring terms as context representation, especially when using large corpora. Additionally, we demonstrate that in our evaluations we obtain the highest scores for all holing operations by computing DTs on large corpora. Comparing different DTs computed with different significance measures to retrieve meaningful *context features* for *language elements*, we demonstrated that the often-used PMI measure does not seem to be well suited for ranking important *context features*. In our experiments we show that the best results are obtained with LL and LMI. Surprisingly, using the frequency of the co-occurrence of *language elements* and *context feature* achieves in some cases the best results.

Our method outperforms two standard methods for computing DTs when using large corpora. Using our pruning strategies does not result to poor performing DTs but results to DTs of higher quality and enables the usage simpler computational methods. Additionally, this allows to compute similarities for all *language elements* within the corpus and is much faster than the two standard methods.

In comparison to word embeddings our DTs, computed with the bigram and trigram holing operation, yield better performances for frequent nouns and comparable results for infrequent nouns when computing the embeddings with standard parameters. Tuning the parameters of the word embeddings, our method scores lower when evaluating the 5 most similar terms against WordNet. However, we observe a better performance using our symbolic approach when we use the 50 most similar terms and beyond. Data analysis reveals that using word

embeddings, the closest similar terms have high similar, but are followed by rare terms, which are either not contained in WordNet or have low WordNet Path scores.

Furthermore, word embeddings compute similarities between words based on numeric vector representations. Thus, no reasoning for the similarities can be provided, as the numeric vectors are hardly interpretable. Using a symbolic approach allows to trace back the context representation, which two terms share. This is important for reasoning, which is mandatory in sensitive applications e.g. in the medical domain cf. Watson Paths (Lally et al., 2014).

Additionally, we have shown that our WordNet based evaluation method for DTs performances comparable to the evaluation against a manually created thesaurus. This supports the usage of the evaluation measure against WordNet, which can be performed automatically.

In another experiment, we have shown the impact of different corpora used for the DT computation. We demonstrated that using Wikipedia achieves similar scores compared to the three times larger newspaper corpus. Also, using the rather small BNC corpus achieves good results, especially for infrequent nouns. This can be attributed to the text quality of this corpus. As Wikipedia is an encyclopedia, it provides better context representations, fitting the nature of a dictionary. In addition, we show the scalability of our approach by computing DTs using large corpora like Google's syntactic n-grams, which are dependency parses from Google Books, and the Googles's, which are n-grams extracted from a large Web corpus. Whereas we achieve the best performance using Google Books, the results based on Web data, which is the largest dataset, achieves low results. The reason for this outcome is the generally lower quality of Web data. Additionally, the thesaurus built on the Web data relies on language-independent preprocessing.

In a last experiment we tested the performance of our method based on verbs. For this, we use the WordNet Path measure and compare our method against results presented by Padró et al. (2014). Except for low frequent verbs, we consistently observe higher evaluation scores with the usage of our DTs. However, they tune the parameters for computing their DT for each frequency band. Based on an evaluation of all verbs we achieve the highest WordNet Path scores by using our method.

We observed that dependency parses are a valuable context feature for building DTs. As supervised dependency parsers are mostly language-dependent, our best performing similarity computation is not language-independent. Based on intrinsic evaluations, unsupervised dependency parsers still perform inferior in comparison to supervised ones. However, no research has been performed where unsupervised parsers are used as context feature for building DTs. Thus, we demonstrate the impact of using unsupervised dependency parses as context representation in order to compute DTs in the next chapter.

Chapter 6

Extrinsic Evaluation of Supervised and Unsupervised Parsers

> You never change things by fighting the existing reality. To change something, build a new model that makes the existing model obsolete.
>
> -Buckminster Fuller

In the previous chapter we have shown the quality of a distributional thesaurus using various context features. The best results are achieved using syntactic collapsed dependency relations generated by the supervised Stanford parser. For the training of supervised parser, manually annotated data is required. As a major goal of this thesis is the extraction of structure in text with knowledge-free and unsupervised methods, we survey whether unsupervised parsers are suited as context representations for computing similarities between *language elements*. This serves as an extrinsic evaluation framework for parsers. Additionally, we can also test whether unsupervised parsers could replace supervised ones. Lastly, we also demonstrate, how information from both can be combined to compute distributional thesauri. This chapter is based on the article published by Riedl et al. (2014a).

6.1 Introduction

Dependency parses give structural and syntactical information about sentences. In many applications within NLP this information has shown to be beneficial e.g. in lexical substitution (see e.g. Section 8.2 and Riedl and Biemann (2013b); Szarvas et al. (2013)), question answering (e.g. Hirschman and Gaizauskas (2001)) or for computing DTs (see e.g. Chapter 3 or Lin (1998); Curran and Moens (2002); Weeds et al. (2004)). But most available parsers are super-

vised parsers. These parsers are trained on so-called treebanks, which are corpora that are manually annotated with syntactic dependencies. The generation of syntactic dependency annotations is time-consuming and thus, only for few languages larger treebanks are available. Furthermore, not only for each language we need a proper treebank but also for each domain. Training a supervised dependency parser on newspaper and applying it to data of a different domain (e.g. Twitter data or medical content) results to a poor performance and domain adaptation is still a non-trivial task (Dredze et al., 2007). Thus, there is a high demand for parsers, which can extract syntactic structure from text unsupervised.

While the field has seen increased interest in automatically inducing syntactic structures from raw or part-of-speech tagged text, the evaluation of unsupervised data-driven parsers has almost exclusively been conducted either by introspection or by automatic comparison to treebanks (e.g. de Marneffe et al. (2006); Bohnet (2010)). It might be due to comparatively low scores on reproducing a treebank's syntactic annotation that hardly anyone has yet attempted to use the output of unsupervised parsers for a NLP task other than parsing itself.

A further complication with unsupervised parsers — be it dependency parsers, constituency parsers or combinatory categorial grammar parsers — is that the categories induced by such parsers cannot be straightforwardly mapped to linguistically-inspired categories as defined in a treebank. Considering only unlabeled syntactic annotations, an unsupervised parser is hardly to blame if it does not adhere to arbitrary conventions. Regarding dependencies for example, it is not a priori clear how to connect auxiliary and main verbs, where to attach the complementizer of subordinate clauses, how to represent a conjunction and its conjuncts, how to relate the preposition and the nominal in prepositional phrases and how to handle punctuation, cf. Nivre and Kübler (2006), Schwartz et al. (2011).

When it comes to *utilizing* syntactic structures, however, it is more important that they are consistent across different sentences than that they adhere to specific syntactic theories and conventions. Here, we choose a task that makes only intermediary use of syntactic structures: we employ unsupervised parsing for preprocessing corpora for the purpose of computing distributional thesauri.

It is generally accepted (e.g. Lin (1997); Curran and Moens (2002), Section 5.7.2) that syntactic preprocessing plays an important role for the quality of DTs. However, comparing words along their syntactic contexts does rely on the existence of such a structure rather than its actual representation. Thus, we believe that distributional similarities are an excellent test bed for addressing the following two research questions: (1) How do unsupervised parsers compare to supervised parsers when used as feature providers for building distributional thesaurus (DT) in comparison to supervised parsers? (2) Can the combination of syntactic parsers increase the quality of a DT?

6.2 Related Work

6.2.1 Evaluating Unsupervised Parsing

As with other unsupervised approaches, the premise of unsupervised induction of syntactic structure is to alleviate the bottleneck of expensive manual annotations for improving NLP applications. For grammar induction, the potential is extremely high due to the complexity of the subject matter: treebanks belong to the most work-intensive NLP datasets. On the other hand, this complexity is hard to grasp for unsupervised systems, which is probably the reason why unsupervised parsing technology is still in its infancy, despite more than a decade of work on this topic.

One of the early works inducing structure from raw sentences and yielding better performance than a random baseline was achieved by van Zaanen (2001), who used an Alignment Based Learning (ABL) approach. This algorithm compares all sentences of a given set and considers matching sequences as constituents. Klein and Manning (2002) presented another approach focusing on constituent sequences called the Constituent-Context Model (CCM). It is an EM-based iterative approach that makes use of the linguistic phenomenon that long constituents often have shorter representations of the same grammatical function that occur in similar contexts. A hybrid approach, combining CCM with a dependency model, called Dependency Model with Valence (DMV) (Klein and Manning, 2004), shows even better performance and is the first unsupervised system to outperform the right branching baseline. Many recent works are based on DMV, such as the system by Headden III et al. (2009), who improved DMV by adding lexical information. Gillenwater et al. (2010) proposed another DMV-based system, which adds a posterior regularization during the training process. Bod (2007) takes a slightly different direction by following an "all subtrees approach", where all possible binary trees are generated for each sentence. The parse of a previously unseen sentence is determined by selecting the most probable tree based on the previously accumulated subtree frequencies. Most of the evaluation of these parsers was performed against a treebank, offering manually annotated and linguistically motivated parse trees. Schwartz et al. (2011) highlight the fact that treebanks contain linguistically problematic annotations, cases without linguistic consensus, such as the decision on the head of a verb phrase or a sequence of nouns. They show that leaving out these cases has a significant but unjustified negative influence of the evaluation outcomes and propose a measure called Neutral Edge Direction (NED), which alleviates this problem. Bod (2007) argues that parser evaluation against a treebank favors supervised approaches and thus measures the parser quality on the outcome of a syntax based Machine Translation (MT) task where the dependency parsers are evaluated as language models. Motazedi et al. (2012) perform an extrinsic evaluation for realization ranking, but only apply a single unsupervised parser and do not compare favorably against a supervised parser. Other extrinsic evaluations with supervised dependency parsers have been performed in informa-

tion extraction systems (Miyao et al., 2008; Buyko and Hahn, 2010) or semantic role labeling (Johansson and Nugues, 2008).

6.2.2 Evaluating Similarities

The similarities of DTs computed with unsupervised parser are evaluated relying on WordNet for English as explained in Section 5.4. For the German evaluation we use the same evaluation method but replace WordNet with GermaNet (Hamp and Feldweg, 1997).

6.3 Methodology

6.3.1 Unsupervised Parsers

In our evaluation, we use five unsupervised parsers, which are described next. They have been selected to span several paradigms of unsupervised syntax induction and due to software availability.

Gillenwater et al. (2010)[54] use a model based on the DMV and improve performance by adding sparsity biases on dependency types. They assume a corpus annotated with POS tags. The aim of this bias is to limit unique head-dependent tag pairs, which is achieved by a constraint on model posteriors during the learning process.

The work of Marecek and Straka (2013)[55] is another enhancement of the DMV and is subsequently referred to as Unsupervised Dependency Parser (UDP). Additionally, it uses prior knowledge in the form of stop estimates that are computed on a large raw corpus using the reducibility principle: a sequence of words is considered as reducible if a word can be removed from the phrase and the remaining part appears another time in the corpus. The assumed property that the first word of a reducible sequence does not have any left children and the last word of this sequence does not have any right children is used for the calculation of such stop estimates. Marecek and Straka (2013) show that estimates computed on a large corpus such as Wikipedia can be used for the parsing of previously unseen text.

Bisk and Hockenmaier (2013) use an expectation–maximization (EM) approach to induce a combinatory categorial grammar (CCG), based on general linguistic assumptions. It creates a model that can be used to parse unseen data. The algorithm requires a corpus, previously assigned with POS tags, in order to be able to distinguish between word classes (mainly to find the verb), and employs general knowledge such as that sentences are headed by verbs. Further language-specific properties are induced from the training data.

[54] http://code.google.com/p/pr-toolkit/

[55] http://ufal.mff.cuni.cz/udp/

Seginer (2007)[56] takes an incremental parsing and learning approach. It operates directly on the plain text without the need for POS tags, by using common cover links (CCL), which can be directly converted to dependency arcs. This parser learns during parsing and can be used without a prior learning step. This should result in increased parsing quality towards later stages, which suggests several passes over the training data. The obtained model can then be reused to parse unseen data. In comparison to the other algorithms, Seginer's parser is not a dependency parser but a constituency parser, which creates parse trees.

The approach of Søgaard (2012) is different from all other approaches discussed here: This algorithm does not require any training data and can operate with or without POS tags. For this reason, it can be applied to arbitrary amounts of data, since it operates sentence-wise without memorizing previous inputs and produces non-projective dependency parses. The words of a phrase are ordered by centrality and a parse is determined by the ranking of a parsing algorithm, which uses general linguistic knowledge for grammar induction. This knowledge is inspired by the rules of Naseem et al. (2010), and the approach has been tuned (once and for all, for all languages) on development data from the Penn Treebank.

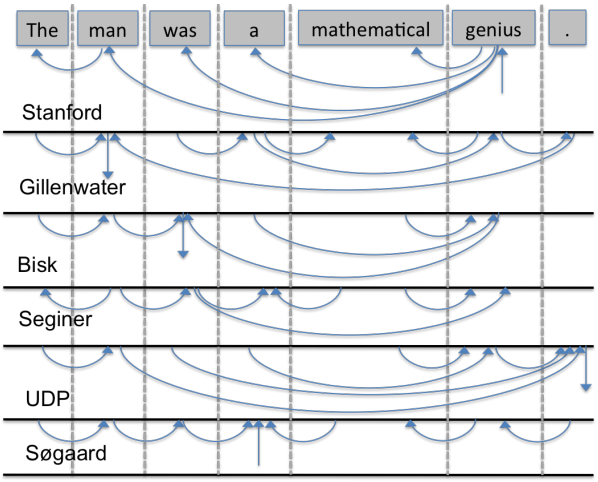


Figure 6.1: Parses for an example sentence for several parsers. Here, Bisk's parser looks most similar to the parses extracted from the Stanford parser. Gillenwater and UDP seem to have some problems with the full stop. Søgaard's parser mostly connects neighbors.

An example sentence is shown in Figure 6.1. The sentences are extracted from the 10M models, except for UDP, where the 1M model is used (cf. Table 6.2 in Section 6.5.1).

Table 6.1 reports the accuracy of four parsers for the English and the German treebanks from the CoNLL-X shared task (Buchholz and Marsi, 2006) predicting unlabeled dependency parses for sentences with length equal and smaller than 10 tokens. Seginer reports only F-

[56] http://www.seggu.net/ccl/

	Baseline	Søgaard	Gillenwater	UDP	Bisk	Seginer	Seginer
English	53.2	59.9	64.4	55.4	70.3	55.6 (WSJ 40)	74.2 (WSJ 10)
German	33.7	57.6	35.7	52.4	68.4	38.2 (Negra 40)	48.0 (Negra 10)

Table 6.1: Unlabeled accuracy values of different unsupervised parsers based on the CoNLL-X shared task (Buchholz and Marsi, 2006). Seginer's results show F-measure values for the Negra and the WSJ corpus. They report results with a maximum sentence length of 10 and 40 words.

scores for WSJ and Negra[57] considering sentences with a maximum length of 10 and 40. The best baselines reported by Canisius et al. (2006) are obtained using a left branching method for English and a nearest neighbor branching method for German. Whereas the nearest neighbor approach is similar to left and right branching, it draws links differently, which refer to verbs and additionally adds links between verbs.

6.3.2 Computing Distributional Thesauri

The extraction of *context features*, used to calculate similarities between terms, is performed with the generic similarity framework proposed in Chapter 5. A (typed or untyped) parser arc is split into term and *context feature*, which consists of the edge direction and label (if any), and the connected term. Similarity between terms is subsequently computed on the overlap of their most salient *context features*.

6.4 Evaluation

We report experimental results on German and English corpora. Both corpora are compiled from 10 million sentences (about 2 Gigawords) each from the Leipzig Corpora Collection[58], randomly sampled from online newspapers. The semantic similarity in English DTs is assessed using WordNet 3.0 as a lexical resource[59]. For evaluating the German DTs, we replace WordNet with its German counterpart, GermaNet 8 (Hamp and Feldweg, 1997). We report results separately for frequent and infrequent selected *language elements* and average the path scores for the most similar 10 words per entry. The English candidate words are the same as used in the previous experiments (see Section 5). For the evaluation of German DTs, we randomly selected 1000 frequent and 1000 infrequent nouns from our German corpus that all occur in GermaNet.

[57] The Negra corpus is available at: `http://www.coli.uni-saarland.de/projects/sfb378/negra-corpus/negra-corpus.html`

[58] `corpora.uni-leipzig.de`, Richter et al. (2006)

[59] As we use WordNet 3.0 instead of WordNet 3.1, which is used Chapter 5, the WordNet Path scores are slightly lower in this chapter.

6.4.1 Experimental Settings

The DTs are calculated using the dependencies from the unsupervised parsers, one at a time. To show the impact of corpus size, we down-sampled our corpora, and used 1 million (1M), 100,000 (100k) and 10,000 (10K) sentences (raw or automatically POS tagged with the TreeTagger[60]) for training/inducing the parsers. Not all parsers were able to deal with the large training sets in feasible runtime, which might be either due to their computational complexity or due to their implementation. While it would be preferable to keep the corpus size for DT computations constant, this was not possible since some unsupervised parsers cannot be applied to unseen text. Hence, we decided to report DT quality results for two setups: Setup A uses the same data for training the parsers and the DT computation. Setup B uses the full corpus of 10M sentences for DT computation, parsed with unsupervised parsers induced on differently sized corpus samples. We feel that Setup B is better reflecting the possible utilization of unsupervised parsers for semantic similarity, since the quality of a DT is known to increase with corpus size (see Section 5.7.5). Nevertheless, we still wanted to assess the quality of parsers that cannot be operated on unseen text in their current state of development.

6.4.2 Four Baselines

We compare the results of unsupervised parsers to four baselines. As a lower-bound baseline, we use a random dependency parser that connects each word in a sentence with a randomly chosen other word. The upper bound is determined by a supervised upper bound baseline. For this, we use Stanford collapsed dependencies (de Marneffe et al., 2006) for the English data and dependencies coming from the Mate tools (Bohnet, 2010) for the German corpus. Finally, to gauge whether the potential of unsupervised parsing to model long-range dependencies — as opposed to local n-gram contexts — lead to better distributional similarities, we use the bigram and trigram holing operation for extracting context (see Section 2.4). The bigram holing operation (see Section 2.4.2) simulates left and right branching and the trigram holing operation shows better similarities, as presented in Section 5.7.2.

We expect the scores of any reasonable unsupervised parser to fall somewhere between the lower bound and the upper bound when compared in the same setting. The n-gram baselines serve as a measure for whether it is worth the trouble to induce and run the unsupervised parser for our evaluation application, as opposed to relying on an arguably simpler system for this purpose.

[60] `www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/`, Schmid (1995)

6.5 Results

In this section, we show results for evaluating DTs built with supervised and unsupervised dependency parsers. First, we show results when computing the DT on German texts and then we present the performance on English texts. Lastly, we evaluate DTs, which have been computed with a combined context representation using supervised and unsupervised parsers.

6.5.1 Single Parser Results for English

We summarize the results for the English evaluation for Setup A and Setup B in Table 6.2. All unsupervised parsers beat the random baseline in all setups, with higher improvements observed using more training data. In addition, more data for DT computation results to higher similarity scores, and rare words generally receive lower scores on average. These finding is expected and validates the DT computation framework.

		10k		100k		1M		10M	
	Parser	freq	rare	freq	rare	freq	rare	freq	rare
Setup A	Random	0.115	0.029	0.128	0.082	0.145	0.103	0.159	0.113
	Trigram	0.133	0.020	0.179	0.082	0.200	0.120	0.236	0.151
	Bigram	0.140	0.029	0.173	0.088	0.208	0.129	0.246	0.159
	Stanford	0.151	0.028	0.209	0.128	0.261	0.176	0.280	0.209
	Seginer	0.136	0.027	0.176	0.085	**0.211**	0.127	0.240	0.155
	Gillenwater	0.135	0.026	0.159	0.077	0.195	0.117	0.223	0.141
	Søgaard	0.120	0.027	0.147	0.083	0.185	0.117	0.227	0.144
	UDP	0.127	0.017	0.169	0.063	0.204	0.119	*	*
	Bisk	0.118	0.017	*	*	*	*	*	*
Setup B	Seginer	0.200	0.063	0.236	0.139	0.241	0.156	0.240	0.155
	Gillenwater	0.220	0.140	0.221	0.142	0.221	0.141	0.223	0.141
	Søgaard	0.227	0.144	0.227	0.144	0.227	0.144	0.227	0.144
	Bisk	0.220	0.139	*	*	*	*	*	*

Table 6.2: Setup A English: Parser induction and DT computation on the same corpus. WordNet Path scores averaged on top 10 similar words, for 1000 frequent and 1000 rare nouns. A * denotes that the evaluation failed because of computational constraints. Setup B English: Parser induction on different corpus sizes, and DT computation on 10M sentences.

In comparison to the n-gram baselines, only the parser by Seginer yields a higher score for frequent words and 1M sentences training in Setup A. Nevertheless, the difference is very small and is confirmed on the 10M sentences only in comparison to the trigram baseline. It seems that Seginer's training procedure saturates somewhere between 100K and 1M sentences, and shows even slightly worse performance on 10M sentences of training in Setup B. None of the parsers seems to be particularly useful as preprocessing steps for DT computations, since better similarities can consistently be reached by *context features* based on bigram statistics.

Comparing the unsupervised parsers, we note that Seginer's approach consistently scores highest in Setup A, while UDP comes in second for frequent words but not for rare words. While Gillenwater's approach reaches comparably high scores for small corpora in Setup A, it is beaten by Søgaard's no-training approach for larger corpora: It seems that Gillenwater's training procedure can hardly make use of additional training, which is shown in Setup B. We observe practically no differences between using 10K and 10M sentences for training the parser. Differences in Setup A are thus solely observed due to increased corpus size for DT computation for the experiments with Gillenwater's parser.

UDP did not finish parsing 10M sentences, after running for 157 days, and it is not trivial to disable its update procedure. Thus, we could not include UDP in Setup B. Bisk's parser is a special case in this evaluation, since it only selects sentences shorter than 15 tokens for training, and hence was effectively trained on a 5400 sentence subset of the 10K corpus. While we did not manage to train it on larger corpora, we could apply this model on 10M sentences in Setup B, where it lands slightly below the no-training parser by Søgaard, but clearly above Seginer's approach for 10K training.

6.5.2 Single Parser Results for German

A different picture is drawn for the German evaluation (see Setup A in Table 6.3). Comparing the results of the unsupervised parsers, Seginer's parser does not only outperform the trigram and bigram baseline for frequent nouns but also the supervised Mate parser for all corpus sizes. Yet, the improvements over the Mate parser are not significant for all results using a paired t-test[61]. Also, Søgaard's parser exceeds the trigram and bigram baseline for 10 million sentences. The remaining unsupervised parsers can beat the random baseline for frequent nouns but none of the n-gram baselines. Again, we are not able to parse the 10 million sentences using UDP and additionally, Gillenwater's parser failed, parsing this corpus. Comparing the baselines in Setup A (see Table 6.3), we observe a difference between the sophisticated baselines and the random baseline only for frequent nouns.

Furthermore, we see that the supervised Mate parser results to worse scores for frequent nouns using the 10k and 100k dataset in comparison to the bigram baseline. This could be attributed to the heavier tail in German's word frequency distribution, which results to sparser *context features* representations for small data[62]. For the 1M and 10M datasets, the supervised parser yields the best similarities for frequent nouns.

The results for Setup B for the German corpora are shown at the bottom in Table 6.3. We observe similar trends to the ones for the English data: using more data for the training does not seem to help the performance of Gillenwater's algorithm. Noticeable is the increase of

[61] Significant improvements ($p < 0.01$) against the Mate parser are marked with the symbol † in Table 6.3 for frequent nouns.

[62] Within the 10M sentences, there are 22 million word types in the German corpus and 10 million word types in the English corpus.

		10k		100k		1M		10M	
	Parser	freq	rare	freq	rare	freq	rare	freq	rare
Setup A	Random	0.097	0.002	0.108	0.010	0.123	0.051	0.143	0.077
	Trigram	0.102	0.002	0.130	0.014	0.159	0.067	0.179	0.086
	Bigram	0.112	0.003	0.130	0.009	0.163	0.053	0.192	0.082
	Mate	0.111	0.004	0.126	0.014	0.170	0.027	0.204	0.090
	Seginer	**0.113**†	0.002	**0.137**†	0.012	**0.171**	0.068	**0.208**	0.091
	Gillenwater	0.104	0.002	0.118	0.009	0.132	0.040	*	*
	Søgaard	0.104	0.002	0.123	0.010	0.161	0.054	0.193	0.077
	UDP	0.107	0.001	0.129	0.004	0.151	0.021	*	*
	Bisk	0.101	0.002	*	*	*	*	*	*
Setup B	Seginer	0.153	0.004	0.186	0.021	0.200	0.092	0.208	0.091
	Gillenwater	0.189	0.080	0.190	0.082	0.189	0.080	*	*
	Søgaard	0.193	0.077	0.193	0.077	0.193	0.077	0.193	0.077
	Bisk	0.185	0.069	*	*	*	*	*	*

Table 6.3: Results for the evaluation based on German corpora. Setup A shows results for various parsers, which are trained and computed based on to different sized corpora. In setup B the parsers are trained on different sized corpora but always applied to 10M sentences, which are then used for computing DTs. The scores present the GermaNet path scores averaged on the most ten similar words from 1000 frequent and 1000 rare German nouns.

Seginer's results for rare words as training data size increases. Seginer's algorithm even beats both n-gram baselines for the 10M corpus when trained only on 1 million sentences.

6.5.3 Combining Parsers for Improving DT Quality

To clarify the best practice for building a DT of high quality, we combine different parsers: the two best-performing unsupervised parsers (Søgaard's and Seginer's), the baselines and the supervised parser. Additionally, these two parsers were the only ones, which could be applied to the largest dataset for both languages.

For English (see Table 6.4), we observe a boost in performance when combining unsupervised parsers. Combining the supervised Stanford parser with the bigram and the trigram baselines also leads to a significant improvement ($p < 0.01$)[63] over the Stanford parser alone, which is about the same as combining the supervised parser with the two unsupervised parsers, and combining all five types of features for DT construction. Overall, a relative improvement of 3.5% on the average WordNet::Path measure for frequent words and a relative 4% improvement for rare words is obtained over the Stanford parser alone.

The results for German (see Table 6.5) show a similar trend. It is remarkable that merging the two unsupervised parsers already outperforms the supervised Mate parser significantly[63] with $p < 0.01$ (6.7% for frequent and 8% relative improvement for rare words). The combination

[63]We use a paired t-test to compare the DTs built using the supervised parser and the combinations.

Parser	frequent	rare
Stanford (supervised)	0.280	0.209
Seginer	0.240	0.155
Søgaard	0.227	0.144
Seginer & Søgaard	0.248	0.162
Stanford & Bigram & Trigram	**0.290†**	**0.217†**
Stanford & Seginer & Søgaard	**0.291†**	**0.217†**
Stanford & Seginer & Søgaard & Bigram & Trigram	**0.290†**	**0.218†**

Table 6.4: Combinations of different parsers for computing English thesauri. The cross (†) indicates significant improvements over the supervised parser. The best results are obtained when combining them all together.

Parser	frequent	rare
Mate (supervised)	0.204	0.090
Seginer	0.208	0.091
Søgaard	0.193	0.077
Seginer & Søgaard	0.218†	0.097†
Mate & Bigram & Trigram	0.204	0.091
Mate & Seginer & Søgaard	**0.222†**	**0.100†**
Mate & Seginer & Søgaard & Bigram & Trigram	**0.222†**	**0.100†**

Table 6.5: Combinations of different parsers for computing German thesauri. Similar to the scores for English (see Table 6.4) the highest scores are obtained when combining all parsers.

of the supervised and unsupervised parsers again leads to further improvement, which is also significant over the supervised parser alone. Adding the bigram and trigram baselines to the three parsers does not give any further improvement.

6.6 Discussion

Overall, it is surprising how well Søgaard's parser performs in comparison to others on this task, since it neither uses training nor relies on POS tags. This hints at either unsupervised parsing being simpler than commonly assumed or rather the immaturity of all unsupervised parsers tested. Further, we would have expected that trained unsupervised parsers, as most unsupervised methods, would exhibit a better performance when trained on larger corpora. This could not be confirmed for both systems that we trained on various corpus sizes, i.e. Seginer's and Gillenwater's approach. The findings are only moderately correlated with evaluations on treebanks, cf. Table 6.1: Whereas Seginer's and Søgaard's parsers perform favorably in our evaluation, they are outperformed by Bisk's parser on treebanks, which currently does not

scale to large data. Gillenwater's parser seems to be overly tuned to English treebanks, but cannot capitalize on this in our evaluation for English.

POS information does not seem beneficial for unsupervised parser induction in noun similarity evaluation, since the highest-scoring approach by Seginer does not use POS tags. Additionally, the version of Søgaard's parser with POS tags scored slightly but consistently lower than the version without POS, as we discovered in further experiments. This is in line with the findings of Cramer (2007), who reports no benefit from manually corrected or unsupervised POS tags for a range of unsupervised parsers.

Comparing the results of previous intrinsic evaluations (see Table 6.1) and the results of our extrinsic evaluation (see Table 6.2 and 6.3), we observe that the ranking of parsers is only mildly correlated. Thus, our proposed evaluation covers different aspects than the adherence to (partially arbitrary) conventions of manually labeled dependency data. In addition, our current evaluation disregards all arcs that do not involve nouns.

When combining parsers, we observe that we can improve the quality of DTs significantly. This leads us to conclude that unsupervised parsers should at least be used for generating features when computing DTs of high quality. In case no high-quality supervised parser is available for the language or domain of interest, it might suffice to use combinations of unsupervised parsers.

We also report the computation times of the different parsers, for the English dataset for Setup A (see Table 6.6). The results were computed on a server with 80 GB of RAM and 16 cores. Whereas all parsers require different amounts of memory, all parsers are single-threaded[64]. While Søgaard's parser is the fastest for small datasets, Seginer's runs faster on 10 million sentences. Whereas Gillenwater's and Seginer's algorithm require almost the same

	10k	100k	1M	10M
Seginer	210	224	261	**508**
Gillenwater	3248	3248	3280	5546
Søgaard	**3**	**21**	**182**	975
UDP	183	1220	11316	-

Table 6.6: Computation time in minutes for parsing the data according to the English corpora used in Setup A, cf. Table 6.2

time for parsing 10k, 100k or 1M sentences, the runtime of the UDP and Søgaard's parser is linear in time with the number of sentences to be parsed. We cannot report the parsing times for Bisk's algorithm, as the parsing was not performed by us. Again, it is noticeable that the best two parsers are also the two unsupervised parsers that run quickest.

[64] As Søgaard's algorithm parses sentence-wise without storing any information it could be easily run multi-threaded.

6.7 Conclusion

The contribution of this chapter is two-fold: First, we have proposed and conducted a comparative extrinsic evaluation procedure for unsupervised parsers based on noun similarity in DTs. Second, we have explored how to improve DT quality by combining features from several parsers. This also proves that our method is able to combine different contextual representations, to acquire the best performances. The transparency of this method with respect to the kind of induced structures (dependencies, constituent trees, combinatory categorial grammar) and with respect to labels of nodes and edges in the parse graph makes it possible to compare different unsupervised parsers without having to rely on treebanks. Since semantic similarity, especially for nouns, is a building block for many NLP applications, we feel that removing the dependency on high-quality supervised parsers can give rise to semantic technologies in many languages.

We have conducted this evaluation with five different unsupervised parsers, and examined the influence of corpus size for parser training and for the similarity computation in a series of experiments. Using established methods for evaluating distributional similarity against lexical semantic resources, we were able to measure differences between parsers in this task that are not reflected by intrinsic evaluations that compare their induced structures to treebanks. These include the influence of corpus size on the training procedure and the consistency of parse fragments on "frequent versus rare words" as well as different languages. Further, we were able to pinpoint a crucial point in unsupervised parsers that has not received much attention: approaches that do not induce an actual parser that can be run on unseen sentences but merely produce syntactic annotations for a given fixed training corpus are hardly useful in applications.

Our evaluation results can be summarized as follows: For English, with its relatively fixed order, Seginer's parser achieves very scarce to no improvements compared to a simple n-gram baseline when used to compute distributional similarities. But for German, Seginer's parser outperforms all baselines including a state-of-the-art supervised parser, and Søgaard's simplistic approach compares favorably to the n-gram baselines. Furthermore, we demonstrate that the quality of noun similarity can be improved significantly when combining the features from supervised and unsupervised parsers.

While today's unsupervised parsers might not be ready for their utilization for semantic similarity for the English language, they can be applied to numerous other languages where highly optimized supervised parsers are not available. Additionally, we feel that our proposed evaluation method exhibits enough sensitivity to be a meaningful test bed for future unsupervised parsers.

Here we have examined the impact of syntactic structure induced by unsupervised parser. Whereas using advanced contextual representation results to similarities, which are more related to lexical resources, also the specification of *language elements* is essential. Aside from

tokens or terms, *language elements* can also be multi-worded expressions, especially as many languages contain these expressions. For this, we will present a method for extracting multi-worded terms from text using distributional semantics in the following chapter.

Chapter 7

Extraction of Multiword Expressions

> Combine two words, Myth and History. What do you get? Mystery.
>
> - David Rains Wallace

In the previous chapters, we have mainly disregarded terms, which consist of more than a single word, called multiword expressions (MWEs). However, MWEs (e.g. *hot dog* or *New York*) are frequently used in many languages. Humans use these terms as single units and are able to recognize that *hot dog* is something to eat and not a dog, which is hot. For a computer the detection of MWEs is a non-trivial task, but an important component in order to understand language. Here, we summarize methods, which rank n-grams according to their multiwordness. Furthermore, we introduce a method, which is based on our hypothesis that terms that are MWEs mostly can be replaced by single-worded terms. This method does not require language-dependent preprocessing and can be applied to textual data in an unsupervised fashion. We show the advantage of our method using a French and an English corpus. Additionally, we report results for differently sized corpora. This chapter is based on the paper published by Riedl and Biemann (2015).

7.1 Introduction

According to Blanc et al. (2007), "language is full of multiword units". Whereas all methods we previously discussed rely on single-worded terms, here we focus on methods for ranking MWEs. By inspecting dictionaries, we can prove the importance of MWEs. For example in WordNet 41.41% of all words are MWEs, as shown in Table 7.1. Whereas more than 50% of all nouns are MWEs, only about 26% of all verbs are MWEs. As the majority of all MWEs found in WordNet are nouns (93.73%), we mainly focus on detecting terms of this particular POS.

	noun	adjective	adverb	verb	all
MWE (count)	60337	505	695	2838	64375
MWE (percentage)	51.51	2.35	15.53	24.59	41.41
all words	117953	21499	4475	11540	155467

Table 7.1: Frequencies and percentages of MWEs contained in WordNet 3.1 for different POS.

While it seems intuitive to treat certain sequences of tokens as single terms, there is still considerable controversy about the definition of what exactly such a multiword expression (MWE) constitutes. Sag et al. (2001) pinpoint the need of an appropriate definition of MWEs. For this, they classify a range of syntactic formations that could form MWEs and define MWEs as being non-compositional with respect to the meaning of their parts. While the exact requirements on MWEs is bound to specific tasks (such as parsing, keyword extraction, etc.), we operationalize the notion of non-compositionality by using distributional semantics and introduce a measure that works well for a range of task-based MWE definitions.

Most previously introduced MWE ranking approaches use the following mechanisms to determine multiwordness: POS tags, word/multiword frequency and significance of co-occurrence of the parts. In this chapter we do not want to introduce "yet another ranking function" but rather an additional mechanism, which performs ranking based on distributional semantics.

Distributional semantics has already been used for MWE identification, but mainly to discriminate between compositional and non-compositional MWEs (Schone and Jurafsky, 2001; Salehi et al., 2014; Hermann and Blunsom, 2014). Here we introduce a concept to describe the multiwordness of a term by its *uniqueness*. Using the *uniqueness* score, we measure how likely a term in context can be replaced by a single word. This measure is motivated by the semiotic consideration that due to parsimony, concepts are often expressed as single words. Furthermore, we implement a context-aware punishment, called *incompleteness*, which degrades the score of terms that seem incomplete regarding their contexts. Both concepts are combined into a single score we call `DRUID`, which is calculated based on a DT. In the following, we show the performance of that method for French and English and examine the effect of corpus size on MWE extraction. Additionally, we report on results without using any linguistic preprocessing except tokenization.

7.2 Related Work

The generation of MWE dictionaries has drawn much attention in the field of NLP. Early computational approaches (e.g. Justeson and Katz (1995)) use POS sequences as MWE extractors. Other approaches relying on word frequency, statistically verify the hypothesis whether the parts of the MWE occur more often together than would be expected by chance (Manning and Schütze, 1999; Evert, 2005; Ramisch, 2012). One of the first measures that consider

context information (co-occurrences) are the C-value and the NC-value introduced by Frantzi et al. (1998). These methods first extract candidates using POS information and then compute scores based on the frequency of the MWE and the frequency of nested MWE candidates. The method described by Wermter and Hahn (2005) computes a score by multiplying the frequency of a candidate when placing wildcards for each word. Lossio-Ventura et al. (2014) introduced a more recent method, which re-ranks scores based on an extension of the C-value and uses POS-based probabilities and inverse document frequencies. Using different measures and learning a classifier that predicts the multiwordness was first proposed by Pecina (2010), who, however, restricts his experiments to two-worded MWEs for the Czech language only. Korkontzelos (2010) comparatively evaluates several MWE ranking measures. The best MWE extractor reported in his work is the scorer by Nakagawa and Mori (2002, 2003), who use the un-nested frequency (called marginal frequency) of each candidate and multiply these by the geometric mean of the distinct neighbor of each word within the candidate.

Distributional semantics is mostly used to detect compositionality of MWEs (Salehi et al., 2014; Katz and Giesbrecht, 2006). For this, most approaches compare the context vector of a MWE with the combined vectors based on the constituent words of the MWE. The similarity between the vectors is then used as degree for compositionality. In machine translation, words are sometimes considered as multiwords if they can be translated as single term (cf. (Bouamor et al., 2012; Anastasiou, 2010)). Whereas this follows the same intuition as our *uniqueness* measure, we do not require any bilingual corpora.

Regarding the evaluation, mostly precision at k ($P@k$) and recall at k ($R@k$) are applied (e.g. (Evert, 2005; Frantzi et al., 1998; Lossio-Ventura et al., 2014)). Another general approach is using the average precision (AP), which is also used in IR (Thater et al., 2009) and has been applied by Ramisch et al. (2012).

7.3 Baselines and Previous Approaches

We evaluate our method by comparing our MWE ranking to multiword lists that have been annotated in corpora. Here, we introduce an upper bound and two baseline methods and give a brief description of the competitive methods used in this chapter. Most of these methods require a list of MWE candidate terms T, usually extracted with POS sequences (see Section 7.5.2).

7.3.1 Upper Bound

We use a perfect ranking as upper bound, where we rank all positive candidates before all negative ones. Within the dataset, we only have binary labels for positive and negative MWEs. Thus, the ranking within the positive and negative MWEs does not influence the upper bound score.

7.3.2 Lower Baseline and Frequency Baseline

The ratio between true candidates and all candidates serves as lower baseline, which is also called baseline precision (Evert, 2008). The second baseline is the frequency baseline, which ranks candidate terms $t \in T$ according to their frequency $freq(t)$.

7.3.3 C-value/NC-value

Frantzi et al. (1998) developed the commonly used C-value (see Equation 7.1). This value is composed of two factors. As first factor, they use the logarithm of the term length in words in order to favor longer MWEs. The second factor is the frequency of the term reduced by the average frequency of all candidate terms T, which nest the term t, i.e. t is a substring of the terms we denote as T_t.

$$\mathrm{cv}(t) = \log_2(|t|)(\mathrm{freq}(t) - \frac{1}{|T_t|}\sum_{b \in T_t}\mathrm{freq}(b)) \tag{7.1}$$

An extension of the C-value was proposed by Frantzi et al. (1998) and is named NC-value. It takes advantage of context words C_t, which are neighboring words of t, by assigning weights to them. As context words only *nouns*, *adjectives* and *verbs* are considered.[65] Context words are weighted with Equation 7.2, where k denotes the number of times the context word $c \in C_t$ occurs with any of the candidate terms. This number is normalized by the number of candidate terms.

$$w(c) = \frac{k}{|T|} \tag{7.2}$$

The NC-value is a weighted sum of the C-value and the product of the term t occurring with each context c, which form the term t_c:

$$\mathrm{nc}(t) = 0.8 * \mathrm{cv}(t) + 0.2 \sum_{c \in C_t} freq(t_c)w(c). \tag{7.3}$$

7.3.4 t-test

The t-test (see e.g. (Manning and Schütze, 1999, p.163)) is a statistical test for the significance of co-occurrence of two words. It relies on the probabilities of the term and its single words. The probability of a word $p(w)$ is defined as the frequency of the term divided by the total

[65] Frantzi et al. (1998) do not specify the context window size.

number of terms of the same length. The *t-test* statistic is computed using Equation 7.4 with *freq*(.) being the total frequency of unigrams.

$$t(w_1 \dots w_n) \approx \frac{p(w_1 \dots w_n) - \prod_{i=1}^{n} p(w_i)}{\sqrt{p(w_1 \dots w_n)/freq(.)}} \tag{7.4}$$

We then use this score to rank the candidate terms.

7.3.5 FGM Score

Nakagawa and Mori (2002, 2003) presented another method that is inspired by the C/NC-value. This method was developed on a Japanese dataset and outperformed a modified C-value[66] measure. It is composed of two scoring mechanisms for the candidate term t as shown in Equation 7.5.

$$FGM(t) = GM(t) \cdot MF(t) \tag{7.5}$$

The first term in the equation is the geometric mean $GM(.)$ of the number of distinct direct left $l(.)$ and right $r(.)$ neighboring words for each single word t_i within t.

$$GM(t) = \sqrt[2|t|]{\sum_{t_i \in t} (|l(t_i)| + 1)(|r(t_i)| + 1)} \tag{7.6}$$

These neighboring words are extracted directly from the corpus; the method does rely on neither candidate lists nor POS tags. In contrast, the marginal frequency $MF(t)$ relies on the candidate list and the underlying corpus. This frequency counts how often the candidate term occurs within the corpus and is not a subset of a candidate. Korkontzelos (2010) showed that while scoring according to Equation 7.5 leads to comparatively good results, it is consistently outperformed by the performance of $MF(t)$.

7.4 Semantic Uniqueness and Incompleteness

We present two mechanisms relying on a DT, which are used to rank terms regarding their multiwordness: A score for the *uniqueness* of a term and a punishing score that conveys the *incompleteness*.

7.4.1 Similarity Computation

The similarities between terms are computed based on the technology introduced in Chapter 5. In order to avoid linguistic preprocessing, we use a *trigram holing operation* considering

[66] They adjust the logarithmic length in order to be able to use the C-value to detect single worded terms.

n-grams (see item *e)* in Section 2.4) up to the length of four.[67] An example for the most similar n-grams to the terms *red blood cell* and *red blood* including the number of features shared with the selected term are presented in Table 7.2.

red blood cell		***red blood***	
Sim. term	Sc.	Sim. term	Sc.
erythrocyte	133	**red**	148
red cell	129	white blood	111
RBC	95	**Sertoli**	93
platelet	70	**Leydig**	92
red-cell	37	**NK**	86
reticulocyte	34	**mast**	85
white blood	33	**granulosa**	81
leukocyte	29	**endothelial**	81
granulocyte	28	hematopoietic stem	79
the erythrocyte	28	peripheral blood monon	78

Table 7.2: We show the ten most similar entries for the term *red blood cell* (left) and *red blood* (right). Here, seven out of ten terms are single words.

7.4.2 Uniqness Computation

The first mechanism of our MWE ranking method is based on the following hypothesis: n-grams, which are MWE, could be substituted by single words, thus they have many single words among their most similar terms. This is motivated by semiotic considerations: Because of parsimony, concepts are usually expressed in single words. When a semantically non-compositional word combination is added to the vocabulary, it expresses a concept that is necessarily similar to other concepts. Hence, multiwordness is indicated if a candidate multiword is similar to many single word terms.

To compute the *uniqueness* score (uq) of an n-gram t, we first extract its most similar n-grams using a DT as described in Section 7.4.1. The function $similarities(t)$ returns the 200 most similar n-grams to the given n-gram t. Using these n-grams, we compute the ratio between unigrams and all similar n-grams using the following formula:

$$uq(t) = \frac{\sum_{s:|s|=1}^{similarities(t)} 1}{|similarities(t)|}. \tag{7.7}$$

We illustrate the computation of our measure based on the MWE *red blood cell* and the non-MWE *red blood*. When considering only the ten most similar entries for both n-grams as illustrated in Table 7.2, we observe a uniqueness score of $7/10 = 0.7$ for both n-grams.

[67] We use LMI for ranking context features and use the default parameters as shown in Table 5.1.

If considering the top 200 similar n-grams that are also used in our experiments we obtain 135 unigrams for the candidate *red blood cell* and 100 unigrams for the n-gram *red blood*. We use these counts for showing the workings of the method in the remainder.

7.4.3 Incompleteness Computation

Similar to the C/NC-value method, we also assign a context weighting function that punishes incomplete terms, which we call *incompleteness (ic)*. For this function, we extract the 1000 most significant context features using the function $context(t)$, which yields to a list of the left context, the hole for the term t and the right context. These context features are the same used for the similarity computation of the DT and have been ranked according to the LMI measure. For the example term *red blood*, some contexts are (*extravasated; @; cells*), (*uninfected; @; cells*), (*nucleated; @; corpuscles*). In the next step we split each list into a tuple representation using the left and right context including its relative position (left/right) to the candidate term. Using the first context feature results in: ⟨*extravasated, left*⟩, ⟨*cells, right*⟩. Then, we sum up the occurrences of for each single context, as shown in Table 7.3 for the two terms.

Context term	Position	Count
red blood cell		
transfusions	right	48
(	right	42
transfusion	right	33
red blood		
cells	right	557
cell	right	344
corpuscles	right	13

Table 7.3: Top three most frequent context words for the term *red blood cell* and *red blood* in the Medline corpus.

We subsequently select the maximal count and normalize it by the counts of considered features $|context(t)|$, which is 1000. This results to the incompleteness measure $ic(t)$. For our example terms we achieve the values $ic(\textit{red blood}) = 557/1000$ and $ic(\textit{red blood cell}) = 48/1000$. Whereas the uniqueness scores for the most similar entries are equal, we now have a measure that signifies the incompleteness of an n-gram with higher scores indicating terms that are incomplete.

7.4.4 Combining Both Measures

As shown in the previous two sections, a high uniqueness score indicates the multiwordness and a high incompleteness score should decrease the overall score. In experiments, we found

the best combination if we subtract[68] the incompleteness score from the uniqueness score. This mechanism is inspired by the C-value and motivated by the fact that terms that are often preceded/followed by the same word do not cover the full multiword expression and need to be downranked. This leads to Equation 7.8, which we call **D**ist**R**ibutional **U**niqueness and **I**ncompleteness **D**egree (DRUID):

$$\texttt{DRUID}(t) = \text{uq}(t) - \text{ic}(t). \tag{7.8}$$

Applying the DRUID score to our example terms (considering the 200 most similar terms) we achieve the scores DRUID(*red blood cell*) = 135/200 − 48/1000 = 0.627 and DRUID(*red blood*) = 100/200 − 557/1000 = −0.057. As a higher DRUID score indicates the multiwordness of an n-gram, we can summarize that the n-gram *red blood cell* is a better MWE than the n-gram *red blood*.

7.5 Experimental Setting

We examine two experimental settings: First, we compute all measures on a small corpus that has been annotated for MWEs, which serves as the gold standard. In the second setting, we compute the measures on a larger in-domain corpus. The evaluation is again performed for the same candidate terms as given by the gold standard. Results for the top k ranked entries are reported using the precision at k ($P@k = \frac{1}{k}\sum_{i=1}^{k} x_i$ with x_i equals 1 if the i-th ranked candidate is annotated as MWE and 0 otherwise). For an overall performance we use the average precision (AP) as defined by Thater et al. (2009): $AP = \frac{1}{|T_{mwe}|}\sum_{k=1}^{|T|} x_k P@k$, with T_{mwe} being the set of positive MWEs. When facing tied scores we mix false and true candidates randomly cf. Cabanac et al. (2010).

7.5.1 Corpora

We consider two annotated (small) corpora and two unannotated (large) corpora for the evaluation and computation of MWEs.

GENIA corpus and SPMRL 2013: French Treebank

In the first experiments, we use two small annotated corpora that serve the gold standard MWEs. We use the medical GENIA corpus (Kim et al., 2003)[69], which consists of 1999 abstracts from Medline[70] and encompasses 0.4 million words. This corpus has annotations regarding

[68] Multiplicative combinations consistently performed worse.

[69] The GENIA corpus is freely available at http://www.nactem.ac.uk/genia/genia-corpus/pos-annotation.

[70] The Medline corpus is available at: http://www.nlm.nih.gov/bsd/licensee/access/medline_pubmed.html.

important and biomedical terms. In addition, single terms are annotated in this dataset, which we ignore.

The second small corpus is based on the French Treebank (Abeillé and Barrier, 2004), which was extended for the SPMRL task (Seddah et al., 2013). This version of the corpus also contains compounds annotated as MWEs. In our experiments we use the training data, which covers 0.4 million words.

Whereas the GENIA MWEs target term matching and medical information retrieval, the SPMRL MWEs mainly focus on improving parsing through compound recognition.

Medline Corpus and Est Républican Corpus (ERC)

In a second experiment, the scalability to larger corpora is tested. For this, we make use of the entire Medline[70] abstracts, which consist of about 1.1 billion words. The Est Républican Corpus (ERC) (Seddah et al., 2012)[71] is our large French corpus. It consists of local French newspapers from the eastern part of France and comprises 150 million words.

7.5.2 Candidate Selection

In the first two experiments, we use POS filters to select candidates. We concentrate on filters that extract noun MWEs, as they constitute the largest amount of MWEs (see Table 7.1) and avoid further preprocessing like lemmatization. We use the filter introduced by Justeson and Katz (1995)[72] for the English medical datasets. Considering only terms that appear more than ten times yields 1,340 candidates for the GENIA dataset and 29,790 candidates for the Medline dataset. According to Table 7.4, we observe that most candidates are bigrams. Whereas for both corpora still around 20% of trigrams are contained, the number of 4-grams is only marginally represented. For the French datasets, we apply the POS filter proposed by Daille et al. (1994)[73], which is suited to match nominal MWEs. Applying the same filtering as for the medical corpora leads to 330 candidate terms for the SPMRL and 7,365 candidate terms for the ERC. Here the ratio between bi- and trigrams is more balanced but again the number of 4-grams constitutes the smallest class.

In comparison to the Medline dataset, the ratio of multiwords extracted by the POS filter on the French corpus is much lower. The reason for that property is that in the French data, many adverbial, prepositional MWEs are annotated, which are not covered by the POS filter.

The third experiment shows the performance of the method in absence of language-specific preprocessing. Thus, we only filter the candidates by frequency and do not make use of POS

[71] The ERC is available at: `http://www.cnrtl.fr/corpus/estrepublicain`.

[72] A regular expression for matching POS tag sequences is summarized by Korkontzelos (2010): `(([JN]+[JN]?[NP]?[JN]?)N)`. Each letter is a truncated POS tag of length one where J is an adjective N a noun and P a preposition.

[73] Following the same convention as for English the regular expression can be expressed as `N[J]?|NN|NPDN`.

Corpus	Candidates	2-gram	3-gram	4-gram
GENIA	1,340	1,056	243	41
Medline	29,790	22,236	6,400	1,154
SPMRL	330	197	116	17
ERC	7,365	3,639	2,889	837

Table 7.4: Number of MWE candidates after filtering for the expected POS tag. Additionally, the table shows the distribution over n-grams with $n \in \{1, 2, 3, 4\}$.

filtering. As most previous methods rely on POS-filtered data, we cannot make use of them in this setting.

For the evaluation, we compute the scores of the competitive methods in two settings: First, we compute the scores based on the full candidate list without any frequency filter and prune low-frequent candidates only for the evaluation (post-prune). In the second setting, we filter candidates according to their frequency before the computation of scores (pre-prune). This leads to differences for context-aware measures, since in the pre-pruned case a lower number of less noisy contexts is used.

7.6 Results

7.6.1 Small Corpora Results

First, we present the results based on the GENIA corpus (see Table 7.5). Almost all competitive

Method	*P*@100	*P*@500	AP
upper baseline	1.000	1.000	1.0000
lower baseline	0.713	0.713	0.7134
frequency	0.790	0.750	0.7468
t-test	0.790	0.750	0.7573
C-value (pre-pruned)	0.880	0.846	0.8447
NC-value (pre-pruned)	0.880	0.840	0.8405
GM	0.590	0.662	0.6740
MF (pre-pruned)	0.920	0.872	0.8761
FGM (pre-pruned)	0.910	0.840	0.8545
MF (post-pruned)	0.900	0.876	0.8866
FGM (post-pruned)	0.900	0.900	0.8948
DRUID	0.930	0.852	0.8663
log(freq)(DRUID)	**0.970**	0.860	0.8661
MF(post-pruned)DRUID	0.950	0.926	**0.9241**
FGM(post-pruned)DRUID	**0.960**	**0.940**	**0.9262**

Table 7.5: This table shows results for P@100, P@500 and the average precision (AP) for various ranking measures. The gold standard is extracted using the GENIA corpus. This corpus is also used for computing the measures.

methods beat the lower baseline. The C/NC-value performs best when the pruning is done

after a frequency filter. In line with the findings of Korkontzelos (2010) and in contrast to Frantzi et al. (1998), the AP of the C-value is slightly higher than for the NC-value. All the FGM based methods except the GM measure alone outperform the C-value. The results in Table 7.5 indicate that the best competitive system is the post-pruned FGM system as it has much higher average precision scores and misses only 50 MWEs in the first 500 entries. A slightly different picture is presented in Figure 7.1 where we plot the *P@k* scores against the number of candidates. Here DRUID performs well for the top-k list for small *k*, i.e. finds many

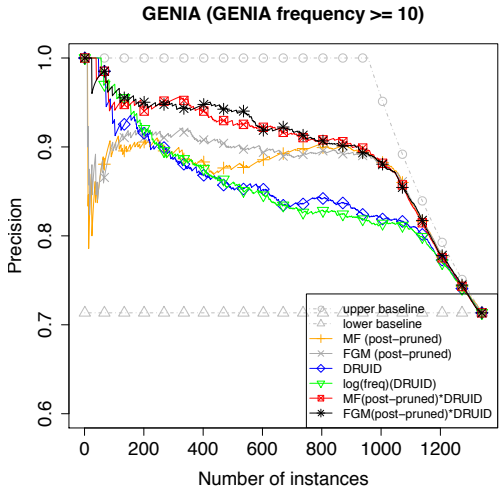


Figure 7.1: This graph shows the P@k for some measures, plotting the precision against *k*. Using DRUID in combination with the MF and FGM measures results to the highest precisions.

valid MWEs with high confidence, thus combines well with MF, which extends to larger k, but places too much importance of frequency when used alone. Common errors are frequent chunks such as "in patience" (we will give more details on errors in Section 7.7). Whereas for the post-pruned case FGM scores higher than MF, the inverse is observed when pre-pruning. Using our DRUID methods can surmount the FGM method only for the first 300 ranked terms (see Figure 7.1 and Table 7.5). Multiplying the logarithmic frequency to the DRUID, the results improve slightly and the best *P*@100 with 0.97 is achieved. All FGM results are outperformed when combining the post-pruned FGM scores with our measure. According to Figure 7.1 this combination achieves high precision for the first ranked candidates and still exploits the good performance of the post-pruned FGM based method for the middle-ranked candidates.

Different results are achieved for the SPMRL dataset as can be seen in Table 7.6. Whereas the pre-pruned C-value again receives better results than frequency, it scores below the lower baseline. In addition, the post-pruned FGM and MF method do not exceed the lower baseline. Data analysis revealed that for the French dataset only ten out of the 330 candidate terms are nested within any of the candidates. This is much lower than the 637 terms nested in the 1340 candidate terms for the GENIA dataset. As both the FGM-based methods and the C/NC-value heavily rely on nested candidates, they cannot profit from the candidates of this dataset

Method	*P*@100	*P*@200	AP
upper baseline	1.000	0.860	1.0000
lower baseline	0.521	0.521	0.5212
frequency	0.500	0.480	0.4876
t-test	0.500	0.485	0.4934
C-value (pre-pruned)	0.490	0.540	0.5107
MF (post-pruned)	0.510	0.495	0.5017
FGM (post-pruned)	0.460	0.480	0.4703
DRUID	**0.790**	**0.690**	**0.7794**
log(freq)DRUID	0.770	0.675	0.7631
MF(post-pruned)DRUID	0.700	0.630	0.6850
FGM(post-pruned)DRUID	0.600	0.570	0.5948

Table 7.6: This table presents the precision French SPMRL corpus. Both the generation of the gold standard and the computations of the measures have been performed on this corpus.

and achieve similar scores as ordering candidates according to their frequency. Comparing the baselines to our scoring method, this time we obtain the best result for DRUID without additional factors. However, multiplying DRUID with MF or log(frequency) still outperforms the other methods and the baselines.

7.6.2 Large Corpora Results

Most MWE evaluations have been performed on rather small corpora. Here we examine the performance of the measures for large corpora, to realistically simulate a situation where the MWEs should be found automatically for an entire domain.

Using the Medline corpus, all methods except the GM score outperform the lower baseline and the frequency baseline (see Table 7.7). Regarding the AP the best results are obtained when combining our DRUID method with the MF, whereas for *P*@100 and *P*@500 the log-frequency weighted DRUID scores best. Using solely the DRUID method or the combined variation with the log-frequency lead to the best ranking for the first 1000 ranked candidates and is then outperformed by the MF based DRUID variations. In this experiment, the C-value achieves the best performance from the competitive methods for the *P*@100 and *P*@500, followed by the t-test. But the highest AP is reached with the post-pruned MF method, which also outperforms the sole DRUID slightly. Contrary to the GENIA results, the MF scores are consistently higher than the FGM scores.

In the French ERC no nested terms are found within the candidates. Thus, the post- and pre-pruned settings are equivalent and thus MF equals frequency. We show the results for the evaluation using the ERC in Table 7.8.

The best results are again obtained with our method with and without the logarithmic frequency weighting. Again, the AP of the C-value and most of the FGM-based methods are inferior to the frequency scoring. Only the t-test and the MF are slightly higher than the

Method	*P*@100	*P*@500	AP
upper baseline	1.000	1.000	1.0000
lower baseline	0.416	0.416	0.4161
frequency	0.720	0.534	0.4331
C-value (pre-pruned)	0.750	0.564	0.4519
t-test	0.720	0.542	0.4483
GM	0.210	0.272	0.3502
MF (pre-pruned)	0.550	0.542	0.4578
FGM (pre-pruned)	0.580	0.478	0.4200
MF (post-pruned)	0.530	0.500	0.4676
FGM (post-pruned)	0.490	0.446	0.4336
DRUID	0.770	0.686	0.4608
log(freq)*DRUID	**0.860**	**0.720**	0.4693
GM*DRUID	0.770	0.634	0.4497
MF(pre-pruned)*DRUID	0.730	0.634	**0.4824**
MF(post-pruned)*DRUID	0.730	0.626	**0.4889**

Table 7.7: This table presents results used on the medical data. Whereas the gold standard is extracted from the GENIA dataset the ranking measures as well as the frequency threshold for selecting the gold candidates are computed using the Medline corpus.

frequency.[74] In contrast to the results based on the smaller SPMRL dataset, the MF, FGM and C-value can outperform the lower baseline. In comparison to the smaller corpora, the performance for the larger corpora is much lower. Especially low-frequent terms in the small corpora that have high frequencies in the larger corpora have not been annotated as MWEs.

7.6.3 Results without POS Filtering

In the last experiment, we apply our method to candidates without any POS filtering and report results using a frequency threshold of ten. As most competitive methods from the previous section rely on POS tags, we only use the t-test for comparison. Analysis revealed that the top-scored candidates according to the t-test begin with stopwords. As an additional heuristic for the t-test, we shift MWEs, which start or end with one of the ten most frequent words, to the last ranks. For the smaller dataset the best results are achieved with the sole DRUID (see Table 7.9) and the frequency weighting does not seem to be beneficial, as highly frequent n-grams ending with stopwords are ranked higher in absence of POS filtering. This, however, is not observed for larger corpora. Here the best results for Medline are achieved with the frequency weighted DRUID. Whereas for French, the sole DRUID method performs best, the difference between the DRUID and the log-frequency-weighted DRUID is rather small. The low APs can be explained by the large number of considered candidates. The second best scores are achieved with the stopword-filtered t-test (t-test + sw). Without a filtered candidate list C-value performs en par with frequency.

[74] This is achieved by chance for the MF, as it is equal to the frequency. The different scores are due to the randomly sorted tied scores used during our evaluation and reflect the variance of the randomness.

Method	*P*@100	*P*@500	AP
upper baseline	1.000	1.000	1.0000
lower baseline	0.220	0.220	0.2201
frequency	0.370	0.354	0.3105
C-value	0.420	0.366	0.3059
t-test	0.390	0.360	0.3134
GM	0.010	0.052	0.1694
MF	0.370	0.356	0.3148
FGM	0.280	0.260	0.2405
DRUID	0.700	0.568	0.3962
log(freq)DRUID	**0.760**	**0.582**	**0.4075**
MF*DRUID	0.570	0.516	0.3776
FGM*DRUID	0.510	0.418	0.3234

Table 7.8: Results for ranking n-grams according to their multiwordness based on the French ERC. The candidates are extracted based on the smaller SPMRL corpus.

7.6.4 Components of DRUID

Here, we show different parameters for DRUID, relying on the English GENIA dataset without POS filtering of MWE candidates and by considering only terms with a frequency of 10 or more. Inspecting the two different components of the DRUID measure (see left graph in Figure 7.2), we observe that the uniqueness measure contributes most to the DRUID score. The main effect of the incompleteness component is the downranking of a rather small number of terms with high uniqueness scores, which improves the overall ranking. We can also see that for

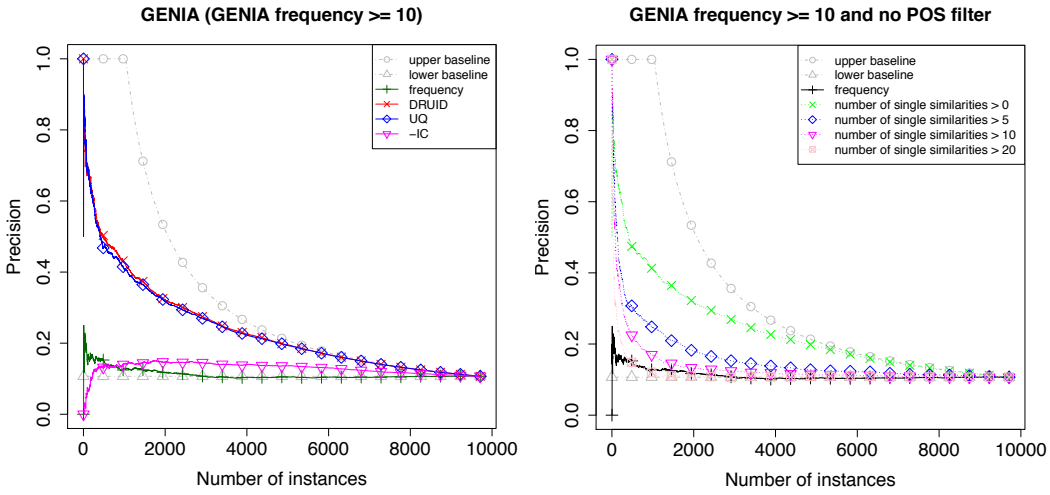


Figure 7.2: Results for the components of the DRUID measure (left) and for different filtering thresholds of the similar entries considered for the uniqueness scoring (rand).

the top ranked terms the negative incompleteness score does not improve over the frequency baseline but outperforms the frequency in the middle ranked candidates. Used in DRUID we

Corpora	Method	Medical		French	
		P@100	AP	*P*@100	AP
small corpora	upper baseline	1.000	1.0000	1.000	1.0000
	lower baseline	0.107	0.1071	0.083	0.0832
	frequency	0.150	0.1135	0.060	0.0906
	t-test	0.160	0.1261	0.080	0.1097
	t-test + sw	0.530	0.3643	0.180	0.1481
	DRUID	**0.700**	**0.4048**	**0.670**	**0.2986**
	log(freq)DRUID	0.690	0.3644	0.460	0.2527
large corpora	upper baseline	1.000	1.0000	1.000	1.0000
	lower baseline	0.036	0.0361	0.019	0.0191
	frequency	0.010	0.0361	0.060	0.0366
	t-test	0.020	0.0412	0.080	0.0440
	t-test + sw	0.000	0.0989	0.200	0.0485
	DRUID	0.610	0.1378	**0.660**	**0.1009**
	log(freq)DRUID	**0.760**	**0.1649**	0.600	0.0988

Table 7.9: This table shows the MWE ranking results based on different methods without using any linguistic preprocessing.

observe a slight improvement for the complete ranking. We achieve a P@500 of 0.474 for the uniqueness scoring and 0.498 for the DRUID score.

When filtering similar entries, used for the *uq* scoring, by their similarity score (see right graph in Figure 7.2), we observe that the amount of similar n-grams considered seems to be more important than the quality of the similar entries: With the increasing filtering also the quality of extracted candidate MWEs diminishes.

7.7 Discussion and Data Analysis

The experiments confirm that our DRUID measure, either weighted with the MF or alone, works best across two languages and across different corpus sizes. It also achieves the best results in absence of POS filtering for candidate term extraction. The optimal weighting of DRUID depends on the nestedness of the MWEs: Using DRUID with the MF should be used when there are more than 20% of nested candidates and using the log-frequency or no frequency weighting when there are almost no nested candidate terms.

We present the best-ranked candidates obtained with our method and with the best competitive method in terms of *P*@100 for the two smaller corpora. Using the GENIA dataset, our log-frequency based DRUID (see left column in Table 7.10) ranks only true MWE within the 15 top-scored candidates.

The right-hand side shows results extracted with the pre-pruned MF method that yields three non-MWE terms. Whereas these terms seem be introduced as candidates due to an POS error, the MF and the C-value are not capable to remove terms starting with stopwords. The DRUID score alleviates this problem with the uniqueness factor. For the French dataset, only

log(freq)DRUID		MF (pre-pruned)	
NF-kappa B	1	T cells	1
transcription factors	1	NF-kappa B	1
transcription factor	1	transcription factors	1
I kappa B alpha	1	activated T cells	1
activated T cells	1	T lymphocytes	1
nuclear factor	1	human monocytes	1
human monocytes	1	I kappa B alpha	1
gene expression	1	nuclear factor	1
T lymphocytes	1	gene expression	1
NF-kappa B activation	1	NF-kappa B activation	1
binding sites	1	in patients	0
MHC class II	1	important role	0
tyrosine phosphorylation	1	binding sites	1
transcriptional activation	1	in B cells	0
nuclear extracts	1	transcriptional activation	1

Table 7.10: Top ranked candidates from the GENIA dataset using our ranking method (left) and the competitive method (right). Each term is marked if it is an MWE (1) or not (0).

one false candidate is ranked in the top 15 candidates. In comparison, eight non-annotated candidates are ranked in the top 15 candidates by the MF (post-pruned) method as shown in Table 7.11.

DRUID		MF (post-pruned)	
hausse des prix	1	milliards de francs	0
mise en oeuvre	1	millions de francs	0
prise de participation	1	Etats - Unis	1
chiffre d' affaires	1	chiffre d' affaires	1
formation professionnelle	1	taux d' intérêt	1
population active	1	milliards de dollars	0
taux d' intérêt	1	millions de dollars	0
politique monétaire	1	Air France	1
Etats - Unis	1	% du capital	0
Réserve fédérale	1	milliard de francse	0
comité d' établissement	1	directeur général	1
projet de loi	1	M. Jean	0
système européen	0	an dernier	1
conseil des ministres	1	années	1
Europe centrale	1	% par rapport	0

Table 7.11: Top ranked candidates from the SPMRL dataset for the best DRUID method (left) and the best competitive method (right). Each term is marked if it is an MWE (1) or not (0).

Whereas the unweighted DRUID method scores better than its competitors on the large corpora, the best results are achieved when using DRUID with frequency-based weights on the smaller corpora. For a direct comparison, we evaluated the small and large corpora using an equal candidate set. We observed that all methods computed on the large corpora achieve

slight inferior results than when computing them using the small corpora. Data analysis revealed that we would consider many of the high ranked "false" candidates as MWEs. For examining the effect, we extracted the top ten ranked terms, which are not annotated as MWE from the methods with the best *P*@100 performance, resulting in the log(freq) DRUID and the pre-pruned C-value methods. We show the terms including their ranking position based on the GENIA dataset in Table 7.12.

log(freq)DRUID		C-value (pre-pr.)	
26	**carboxylic acid**	1	present study
28	**connective tissue**	7	important role
40	**cathepsin B**	11	degrees C
41	**soft tissue**	13	**risk factors**
42	**transferrin receptor**	15	significant differences
53	DNA damaging	18	other hand
61	**foreign body**	22	significant difference
62	radical scavenging	33	**magnetic resonance**
71	spatial distribution	39	first time
74	**myosin heavy chain**	48	significant increase

Table 7.12: Top ranked terms for the Medline corpus, which are not marked as MWEs. The rank is denoted left to each term and all terms, which can be found within a lexicon, are marked bold.

First, we observe that the first 'false' candidate for our method appears at rank 26 and at rank 1 for the C-value. Additionally, only ten out of the top 74 candidates are not annotated as MWEs for our method and 48 for the competitor. When searching the terms within the MeSH dictionary[75], we find seven terms ranked from our method and two for the competitive method.

In contrast to the competitive measures introduced in this chapter, our method is also able to rank single-worded term. We show the 20 highest ranked single-worded terms in Table 7.13 for the Medline and the ERC corpus. In both lists we did not filter by POS and removed numbers, which often have a high DRUID score. Both for French and for the medical data we observe some verbs, but also a lot of content words. These might be well suited as keyword lists in e.g. search engines or automatic speech recognition.

7.8 Conclusion

Uniqueness is a well-working mechanism in MWE modeling. Whereas frequency and co-occurrence have been captured in many previous approaches (see Manning and Schütze (1999), Ramisch et al. (2012) and Korkontzelos (2010) for a survey), we boost multiword candidates *t* by their grade of distributional similarity with single word terms. We implement such contextual substitutability with a model where the term *t* can consist of multiword tokens and similarity

[75]The MeSH dictionary is available at: http://www.nlm.nih.gov/mesh/.

Medline		ERC	
GATA-1	antiatherosclerotic	mesure	Bergé
function	Smad6	activités	carnets
Sp1	Evi-1	politique	Bouvet
used	ETS1	prix	promesse
increased	3q26	réduction	préoccupe
shown	Tcf	analyse	composants
IFN-gamma	LEF-1	crise	aspirations
decreased	hypolipidaemic	stratégie	hostilité
IL-10	down-regulatory	tête	dettes
IL-5	Xq13	campagne	Brunet

Table 7.13: This table shows the highest ranked single-worded terms for Medline and ERC without any POS filtering based on the DRUID score.

is measured based on the right and neighboring word between all (single and multiword) terms. Since it is the default to express concepts with single words, a high uniqueness score is given to multiwords that belong to a category just as single words would. For example using an English open-domain corpus, *hot dog* is most similar to the terms: *food*, *burger*, *hamburger*, *sausage* and *roadside*. Candidates with a low number of single word similarities also serve the same function, but more frequently we observe single n-grams with function words or modifying adjectives concatenated with content words, e.g. *small dog* is most similar to "*various cat*", "*large amount of*", "*large dog*", "*certain dog*", "*dog*". To be able to kick in, the measure requires a certain minimum frequency for candidates in order to find enough contextual overlap with other terms. Additionally, we demonstrate effective performance on larger corpora and show its applicability when used in a complete unsupervised evaluation setting.

Chapter 8

Applying Distributional Thesauri

> Knowledge without application is like a book that is never read.
>
> – Christopher Crawford

> It is always more easy to discover and proclaim general principles than to apply them.
>
> - Winston S. Churchill, in *The Gathering Storm*

In Chapter 5 we have introduced a framework for computing semantic similarities between words. However, we have only evaluated them against lexical resources. In this chapter, we inspect the benefit of our semantic similarities when plugging them as features into supervised systems. We exemplify the usefulness of our resources based on three different NLP tasks. First, we present a general approach for replacing unknown words when applying a supervised classifier to previously unseen text. The replacement is performed using a DT. We evaluate the performance of this approach using an existing POS tagger. The second task is an adaptation of a lexical substitution algorithm to the medical domain. This algorithm replaces terms in context with terms of the same meaning. We can show that using a DT computed on medical data is one of the most prominent features for this task. In the third section we introduce a supersense clustering, which is computed using LDA and is based on a DT. We show the benefit of this feature as well as features from a DT based on a Named Entity Recognition (NER) task.

8.1 Part-of-Speech Tagging for Out-of-Vocabulary Words

> There are ten parts of speech and
> they are all troublesome
>
> - Mark Twain, in *The Awful German Language*

For supervised learning methods in NLP it is mostly easy to predict structure for words they have already seen during the training. However, the prediction of structure is much harder for unknown words. But, according to the power-law distribution (see Section 2.6), the majority of words are infrequent and thus the number of unknown words might be high. Supervised systems aim to retrieve a general representation of words, so they can also be applied to words that have not been in the vocabulary during the training time. These words are called out-of-vocabulary (OOV) words. But the best performance is mostly achieved for words, which have been seen during the training of the classifier. In this section, we examine the problem of the OOV words and investigate whether the performance improves when we replace OOV words with words known during training time using a distributional thesaurus (DT). Whereas such a replacement can be plugged in to most NLP tasks, we investigate the effect of OOV words based on the task of POS tagging. While POS tagging is generally regarded as a solved problem for languages and domains with sufficient amounts of training data, there are still challenges in domain adaptation, e.g. for user-generated content (Gimpel et al., 2011) or for domain-specific texts. Biemann (2009) reports a rate of 20% OOV words for a medical corpus (Medline abstracts) when considering the vocabulary of a general corpus (BNC). The largest source of errors in POS assignment is observed for OOV words, as they can only be classified using contextual and surface features. A sequence of OOV words can throw off the sequence classification algorithm, resulting in poor performance. For classifiers that do not normalize over the whole sequence, this has been described as the label bias problem (cf. Lafferty et al. (2001)). Parts of this section are based on Section 2.3 in (Biemann and Riedl, 2013).

8.1.1 Method

We train a POS tagger based on training data and construct a dictionary of words contained in the training data.[76] Using a larger background corpus, we compute a DT as described in Chapter 5. When applying the POS tagger to unseen text, we check whether all words are contained in the dictionary. If an OOV word is found, we look up the similar words in the DT and replace it with a word that is contained in the vocabulary. For the replacement, two

[76] Most POS taggers use regular expressions to detect e.g. numbers and use heuristics for capitalization. We show the advance of replacing different kinds of "unknown" words in the result section.

approaches are considered. The first one replaces the word with the most similar word, which is contained in the dictionary.

In the second approach, we replace a word with the most similar word, which is known and has the highest suffix overlap with the one it replaces. This has shown to be helpful in (Ratnaparkhi, 1996; Toutanova et al., 2003) and might be beneficial especially for POS tagging as suffixes are morphological markers in many languages. After the replacement, the POS tagger is applied to the modified sentence. The predicted POS tags are then projected onto the original text.

Figure 8.1 illustrates this method using an example: In the sentence "Renting out an unfurnished two-bedroom triplex in San Francisco", the words "unfurnished", "one-bedroom" and "triplex" are OOV words, not being part of the training set (marked red). The correct POS tags for these three words are adjective (*JJ*), adjective (*JJ*) and noun (*NN*). It might be

Figure 8.1: Illustrating the two-dimensional extension for POS tagging. The red-marked words are unknown and we show the expansions from a DT. Additionally, we mark unknown words with their appropriate POS tags. According to the tagset used in the PTB adjectives are marked as *JJ* and nouns as *NN*.

surprising that the word "one-bedroom" is unknown, but within the PTB, which is used for the evaluation, a spelling without the dash is used, resulting in two tokens "two bedroom". While the top-most similar words to "unfurnished" and "triplex" ("empty" and "duplex") are in-vocabulary words, the most similar in-vocabulary word for "one-bedroom", "two-room", is the third most similar expansion according to our DT. Tagging the alternate sentence "Renting out an empty two-room duplex in San Francisco" results in correct assignment of POS tags.

8.1.2 Experimental Setting

The impact of our method is evaluated using the TreeTagger (Schmid, 1994, 1995) for POS tagging without considering probabilities for unknown tokens.[77] For a language-independent approach, we use the trigram holing operation to compute a DT as exemplified with *c)* in Section 2.4.

[77] We use the parameters `-base` for using only probabilities for known tokens and enable the options `-hypen-heuristics`, `-ignore-prefix`, `-cap-heuristics`.

The evaluation is performed both for German and for English. For English we use the Wall Street Journal (WSJ) collection from the Penn Treebank (PTB, Marcus et al. (1993)) and follow Collins (2002) by training on sections 0–18 and testing on sections 22–24.[78] We use the 105M sentences from the newspaper corpus (see Section 5.6) for the computation of the DT.

For the evaluation based on German, we use the TIGER Treebank (Brants et al., 2002). We follow Giesbrecht and Evert (2009) and use a 10-fold cross-validation (CV) for the evaluation. The DT is computed using 70M sentences of German newspapers selected from the LCC. A subset of this corpus has also been used in Chapter 6.

8.1.3 Results

When we evaluate the impact of replacing unknown tokens, these tokens can be classified in different types of OOV tokens. It would be obvious to refer to a token as OOV if it has not been seen in the training data. We use a simple POS tagger and do not use any smoothing strategies included in TreeTagger for unknown tokens. However, TreeTagger still recognizes terms that are not contained in the lexicon. These terms are mostly number and capitalized words. Thus, we show the performance based on different considerations of unknown words. Words, which are not in the training data, are abbreviated with U+R. These words can be split into words, which can be replaced using the DT (R) and words that cannot be replaced (U) as they are not contained within the corpus used for building the DT. Furthermore, TreeTagger features the functionality to mark words as unknown. Mostly it does not mark words as unknown that are numbers and words that have a different casing than the known word. We also report on the performance solely of these words (TT). Additionally, we show results based on all words (all).

We show the number of unknown and all tokens for both datasets in Table 8.1 for the test set. For the WSJ dataset we observe 7691 tokens, which are not contained in the training data (see columns two and three considering U+R in Table 8.1) and around 80%[79] of these tokens can be replaced with our DT. But only around 46%[80] of these tokens are considered as unknown for the TreeTagger. Data analysis reveals that most of the additional unknown tokens are numbers. To avoid replacing numeric numbers, which are known to the TreeTagger, we use a regular expression[81] and consider the matching tokens as known tokens, as shown in columns four and five. Now, the remaining difference between the tokens in U+R and TT are mainly proper nouns and common nouns.

The TIGER dataset does not contain any numeric values. Thus, filtering numeric values does not change the data. We show floating point numbers as the numbers are computed

[78] We do not perform parameter optimization and thus do not use section 19–21, which is normally used for development

[79] R/(U+R)=6204/7691

[80] TT/(U+R)=3570/7691

[81] We use the following regular expression to detect numeric tokens: `([-]?[0-9]*[.,:-][0-9]+)`.

	WSJ				TIGER	
	all tokens		non-numeric tokens		all tokens	
	number	percentage	number	percentage	number	percentage
all	171138	100.00%	171138	100.00%	34728.10	100.00%
U	1487	0.87%	317	0.19%	296.40	0.85%
R	6204	3.63%	3724	2.18%	2657.20	7.65%
U+R	7691	4.49%	4041	2.36%	2953.60	8.50%
TT	3570	2.09%	3570	2.09%	2837.40	8.17%

Table 8.1: This table shows the number and percentage of unknown tokens for the English WSJ based on the test set in comparison to the training data. We also show numbers for the TIGER dataset based on a 10-fold CV. We show numbers for all tokens and for tokens, which are not in the training data (U+R) and additionally show the numbers for tokens not in the training data, which cannot be replaced using the DT (U) and for tokens, which can be replaced using the DT (R). The last row shows the numbers for tokens, which are unknown regarding the TreeTagger (TT). TreeTagger mostly is able to identify numbers. Thus, we also show numbers when not considering unknown numbers as unknown tokens using a regular expression. In the TIGER dataset no numbers are contained according to our number detector.

based on the 10-fold CV on the test sets divided by ten. The corpus is much smaller than the WSJ but in relation to the English dataset the number of unknown tokens is much higher. Using this dataset also the number of unknown words compared to the dictionary (U+R) and the unknown tokens marked by Treetagger (TT) are very similar.

First, we present the results using the English dataset for the different subsets as introduced above. For each subset (U, R, U+R, TT, all) we report on the accuracy when not replacing tokens (no replacement) when replacing unknown tokens that are not in the training data (replace R), and when replacing only tokens, which TreeTagger marks as unknown (replace TT). We first show the results in Table 8.2 using no numeric filter and replacing unknown words with the most similar word found in the DT, which is contained in the list of known words.

	U	R	U+R	TT	ALL
no replacement	**0.8796**	0.7045	0.7384	0.5022	0.90221
replace R	**0.8796**	0.5459	0.6105	**0.7014**	0.89646
replace TT	**0.8796**	**0.8138**	**0.8266**	0.6924	**0.90618**

Table 8.2: Accuracy for the TreeTagger based on the WSJ dataset without filtering numeric values and replacing unknown tokens by most similar entry from the DT, which is contained within the dictionary. A listing of the different abbreviations is explained in Table 8.1.

We observe that the accuracy does not change for unknown tokens, which cannot be replaced (U). Whereas this might be expected, we also might expect improved results, especially when replacing unknown tokens next to the token, which cannot be replaced, as this gives the sequence classifier more evidence by the context. Data analysis reveals that none of the

tokens that cannot be found within the DT (U) have unknown words in its neighborhood, which cannot be replaced. Interestingly, we observe a poor performance when replacing tokens not found within the training data, which is shown in the third row (replace R) and third and fourth columns (R and U+R) in Table 8.2. In addition, the overall performance (column all) declines slightly. However, replacing only words that are unknown to the TreeTagger (column TT) improves the performance of these unknown words by 0.2. Using this replacement, we also gain slight improvements when we apply the evaluation to all words. Due to the poor performance when replacing unknown words not in the training data, we analyze the replaced words, which are classified wrongly. Most of these words are numbers, which are replaced with known tokens in the training data but mostly have different POS. For example *4:30* is replaced with *noon* and year dates are sometimes replaced with month names.

In order to avoid the bias introduced by replacing numeric tokens, we perform a second experiment. In this experiment, we consider unknown numeric tokens as known tokens, using the regular expression described in footnote [81], and do not replace them. Comparing the results shown in Table 8.3 with the previous results in Table 8.2, we observe a drop in the performance when no replacement is performed for the evaluation based on unknown tokens (U, R, U+R and TT). This attests that the correct classification of numbers (known and unknown) seems

	U	R	U+R	TT	all
no replacement	**0.4416**	0.5099	0.5046	0.5022	0.90221
replace R	**0.4416**	**0.6950**	**0.6751**	**0.7014**	**0.90623**
replace TT	**0.4416**	0.6920	0.6724	0.6924	0.90618

Table 8.3: This table shows the accuracies for the WSJ dataset when not considering numeric values as unknown and replacing them by similar words. Again, we show results with no replacements of OOV terms, replacing terms, which have not been within the training data (R) except numbers and terms, which are unknown to the TreeTagger (TT). Based on this selection we show the performance for terms unknown within the DT (U), unknown terms, which can be replaced with the DT (R) their combination and for the terms, which are unknown according to the TreeTagger (TT).

to work rather well for TreeTagger, but the correct classification of remaining unknown words remains difficult. Now we observe the best results when replacing the tokens, which are not within the training data (replace R). Whereas we achieve tremendous improvements up to 0.2 for the unknown tokens (R, U+R and TT) the improvement for all tokens is only marginal, as the amount of unknown tokens is rather small. Furthermore, we also achieve better results using our numeric filter than when replacing the words that TreeTagger marks as unknown.

Next, we extend the strategy for selecting the substitute from the DT and replace unknown tokens with tokens within the DT, which have a high suffix overlap to the unknown word, as explained in Section 8.1.1. Again, we treat numeric tokens as known ones. This time we observe further improvements (see Table 8.4). Considering the results achieved when replacing tokens, except numbers, with known ones (replace R) we observe improvements up to 0.27 for

replaceable terms (R). Additionally, the performance for all tokens improves. As observed in

	U	R	U+R	TT	ALL
no replacement	**0.4416**	0.5099	0.5046	0.5022	0.9022
replace R	**0.4416**	**0.7801**	**0.7535**	**0.7770**	**0.9081**
replace TT	**0.4416**	0.7519	0.7275	0.7549	0.9075

Table 8.4: Results for the WSJ corpus when avoiding the replacement of unknown numeric values and replacing words with the most similar one sharing the highest suffix overlap with the unknown word.

the previous experiment, our method for marking numbers as known tokens (replace R) results in a better performance compared to the results achieved when replacing only unknown tokens marked by TreeTagger (replace TT). This results in the recommendation to replace all tokens, which are not contained in the training data, except numeric tokens, using a simple regular expression.

When using the POS tagger in a more sophisticated way by using probabilities for unseen tokens[82], we observe an accuracy of 0.9574 without replacing OOV tokens. For avoiding unknown tokens, we use the suffix based replacement strategy, which performed best. Replacing tokens, which are marked as unknown using the TreeTagger (replace TT), results to an accuracy of 0.9596. Slightly better results are accomplished by replacing unknown non-numerical tokens, which results to an accuracy of 0.9599.

Next, we present the results based on the German TIGER dataset using a 10-fold CV. We first show the accuracies when replacing unknown tokens with the most similar words from the DT, which are contained in the training data (see Table 8.5). Again, the results do not change for unknown words, which are not contained in the DT (U) when replacing the remaining unknown words (R). Similar to the results for English, we achieve better results when

	U	R	U+R	TT	ALL
no replacement	**0.7042**	0.6418	0.6481	0.6485	0.8426
replace R	**0.7042**	**0.8215**	**0.8097**	**0.8187**	**0.8563**
replace TT	**0.7042**	0.7838	0.7758	0.7815	0.8534

Table 8.5: Accuracies based on the TIGER dataset by replacing unknown tokens by the most similar word from the DT, which is contained in the dictionary.

replacing tokens, which are not in the lexicon (replace R), in comparison the results when replacing words that are marked as unknown by the TreeTagger (replace TT). The highest improvement is observed for unknown words, which can be replaced by the DT (R). Replacing tokens not in the training data (replace R) increases the accuracy from 0.6418 up to 0.8215. Additionally, the performance improves for all words from 0.8426 up to 0.8563.

[82] We disable the *-base* parameter.

Next, we show the accuracy results when using the second replacement strategy: we replace an unknown token by the most similar token from the DT, which shares the longest suffix. Using this strategy boosts the performance again as shown in Table 8.6. Comparing

	U	R	U+R	TT	ALL
no replacement	**0.7042**	0.6418	0.6481	0.6485	0.8426
replace R	**0.7042**	**0.8628**	**0.8468**	**0.8549**	**0.8595**
replace TT	**0.7042**	0.8214	0.8096	0.8167	0.8563

Table 8.6: Results for the TIGER dataset when replacing unknown words by the most similar word with highest suffix overlap with the replaced word.

the simple DT replacement to the suffix-based replacement the accuracy increases from 0.8097 to 0.8468 when replacing and evaluating tokens, which are not in the training data (row: replace R, column: R) Again, the accuracy for all tokens increases slightly in comparison to the results in Table 8.5. Furthermore, our strategy on deciding on unknown tokens turns out to be better than the method used within TreeTagger.

Applying the TreeTagger with smoothing strategies for unknown tokens, we achieve an accuracy of 0.9630 without any replacement and 0.9668 when replacing tokens not in the training data using the suffix-based DT replacement. In line with the English results, replacing tokens, which are unknown to the TreeTagger, results in a lower accuracy of 0.9643.

The overall performance we achieve is below the state-of-the-art results for POS tagging on this dataset as shown in Table 8.7. For the English WSJ dataset our method improves the

Method	**English**	**German**
TreeTagger	0.9574	0.9630
TreeTagger + OOV replacement	0.9599	0.9668
Best results	**0.9750**	**0.9763**

Table 8.7: This table shows the overall performance for the English and German dataset based on the test set. We show results for the TreeTagger with and without replacing OOV words as well as the best performing systems, which have been proposed by Søgaard (2011) for English and by Giesbrecht and Evert (2009) for German.

accuracy from 0.9574 to 0.9599. However, the best performing system proposed by Søgaard (2011) reaches an accuracy of 0.9750. Based on the German dataset we improve from 0.9630 to 0.9668. The best performing system by Giesbrecht and Evert (2009) achieves an accuracy of 0.9763. But in contrast to TreeTagger and our approach, these approaches make heavy use of surface feature, backoff, word clustering on background corpora, and advanced machine learning techniques. Our setup, illustrates how a DT can be used in the context of existing NLP tools, with neither needing to alter the feature representation nor the machine learning algorithm. The novelty with respect to word-space approaches is that the DT is able to generate

the most similar tokens, so that they can be used in lieu of tokens that impose difficulties for the software (i.e. OOV words for POS tagging). A comparable approach of expanding text representations with similar words was successfully used by Miller et al. (2012) for knowledge-based word sense disambiguation.

8.1.4 Conclusion

In this section, we introduced a method for tackling the OOV problem, by replacing unknown words using a DT. We showed the impact of this method for the task of POS tagging. For this we used the TreeTagger and presented the performance for German and English. We demonstrated performance gains on both languages for OOV words and observed the best results when replacing non-numeric words, which have been unseen in the training data, using the most similar word from a DT with the longest suffix overlap with the unknown word. Whereas we observe large improvements for the OOV words, we also observe performance gains for the overall performance. Our method is not only suited for the task of POS tagging, but can also be applied to different tasks e.g. machine translation (Gangadharaiah et al., 2010; Zhang et al., 2012) or dependency parsing (Chen et al., 2014).

8.2 Lexical Substitution for the Medical Domain

> The change of the word does not alter the matter
>
> - Thomas More, in *Utopia*

The task of lexical substitution (McCarthy and Navigli, 2009) deals with the substitution of a *target term* within a sentence with words of the same meaning. Exemplary we show a sentence in Figure 8.2 with the target term *bugle*. For replacing a term, we require a list of

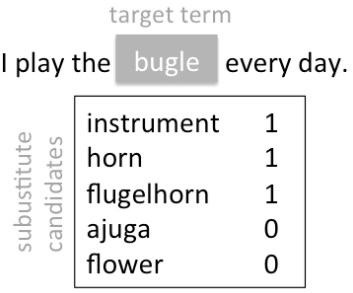


Figure 8.2: Example sentence, where the target term *bugle* should be replaced by a substitute term fitting into the context. The list of substitutes contains annotations whether they fit (1) into the context or not (0).

substitute candidates. In the example five terms are provided including annotations indicating if a candidate fits into the context (1) or not (0). From the context (here the word *play*), a human recognizes that the term *bugle* is used in the context of a music instrument rather than the plant sense.

The task of lexical substitution is divided into two subtasks:

- Identification of *substitution candidates*, e.g. terms that are, for some contexts, substitutable for a given target term.
- Ranking the substitution candidates according to their context

Having such an automatic substitution system can help for semantic text similarity (Bär et al., 2012), textual entailment (Dagan et al., 2013) or plagiarism detection (Chong and Specia, 2011).

The datasets, provided by McCarthy and Navigli (2009) and Biemann (2012), offer manually annotated substitutes for a given set of target words within a context (sentence). Contrary to these two datasets Kremer et al. (2014) introduced a dataset where all words are annotated with substitutes. All these datasets are suited for the open domain.

A system performing lexical substitution is not only of interest for the open domain, but also for other domains e.g. the medical domain. Such a system can then be applied to medical word sense disambiguation, entailment or question answering tasks. Here, we use a medical dataset described by (Riedl et al., 2014b) and adapt the lexical substitution system, provided by Szarvas et al. (2013), to the medical domain. Additionally, we do not use WordNet (Miller, 1995) to provide similar terms, but rather employ a DT, computed on medical texts. This section is based on the article by Riedl et al. (2014b).

8.2.1 Related Work

For the general domain, the lexical substitution task was initiated by Semeval-2007 Task #10 (McCarthy and Navigli, 2009). In this task, the best performance was achieved by an unsupervised method (Giuliano et al., 2007), which uses WordNet for the substitution candidate generation and then relies on the Google Web1T n-grams (Brants and Franz, 2006) to rank the substitutes.

To our knowledge, Szarvas et al. (2013) propose the currently best system. This is a supervised approach, where a single classifier is trained using delexicalized features for all substitutes. Using such features, the classifier can be applied even to previously unseen target terms and substitutes. Although there have been many approaches for solving the task for the general domain, only slight effort has been done in adapting it to different domains.

8.2.2 Method

To perform lexical substitution, we follow the delexicalization framework of Szarvas et al. (2013). Delexicalization focuses on making features independent of the word they are based on. For the lexical substitution task for each candidate, several substitutes might be possible. To delexicalize features for a substitute they are designed to be relative to the target term. For instance when using the frequency of the substitute, it is divided by the frequency of the target term.

We automatically build DTs for the medical domain and use features from the Unified Medical Language System (UMLS) ontology. The dataset for supervised lexical substitution consists of sentences, containing an annotated target word t. Considering the sentence being the context for the target word, the target word might have different meanings. Thus, correct substitute candidates $s_{g_1} \dots s_{g_n} \in s_g$, need to be provided for each context. Negative examples are substitute candidates that are either incorrect for the target word, do not fit into the context or both. We refer to these substitutes as *false substitute candidates* $s_{f_1} \dots s_{f_m} \in s_f$ with $s_f \cap s_g = \emptyset$. Both substitute sets form the amount of all substitutes $s = s_g \cup s_f$.

For the generation of substitute candidates we do not use WordNet, as done in previous works (Szarvas et al., 2013), but use only substitutes from a DT. To train a single classifier, features that distinguishing the meaning of words in different context need to be considered. Such features could be e.g. n-grams, features from distributional semantics or features, which are extracted relative to the target word, such as the ratio between frequencies of the substitute candidate and the target word. After training, we apply the algorithm to unseen substitute candidates and rank them according to their positive probabilities, given by the classifier. Contrary to Szarvas et al. (2013), we do not use any weighting in the training if many annotators selected the same term, as we could not observe any improvements. Additionally, we use logistic regression (Fan et al., 2008)[83] as classifier.

8.2.3 Resources

For the substitutes and for the generation of delexicalized features, we rely on DTs, the UMLS and Google Web1T.

Distributional thesauri (DTs)

We computed two different DTs using the framework proposed in Chapter 5.[84]

[83] We use a Java port of LIBLINEAR (`http://www.csie.ntu.edu.tw/~cjlin/liblinear/`) available from: `http://liblinear.bwaldvogel.de/`

[84] We use LMI (Bordag, 2008) as significance measure and consider only the top 1000 ($p = 1000$) features per term.

The first DT is computed based on Medline[85] abstracts. This thesaurus uses the left and the right word as context features (called *n*-gram trigram holing operation as described in Section 2.4). To include multiword expressions, we allow the number of tokens that form a term to be up to the length of three.

The second DT is based on dependencies as context features from a English Slot Grammar (ESG) parser (McCord et al., 2012) modified to handle medical data. The ESG parser also detects multiword expressions using a predefined list extracted from dictionaries and the UMLS. As input data, we use 3.3 GB of texts from medical textbooks, encyclopedias and clinical reference material as well as selected journals. This DT is also used for the generation of candidates supplied to annotators when creating the gold standard and is the main resource to provide substitute candidates. We can observe reasonable similarity entries for both DTs in Table 8.8, which illustrates the top entries for the term *mass spectrometry*.

ESG Parser	**Medline Trigram**
hplc	tandem mass spectrometry
flow cytometry	ms
mass spectrometer	mass spectrometer
fluorescence in situ hybridization	hplc
mri	mass spectrometric
tandem mass spectrometry	lc
real time per	ionization mass spectrometry
elisa	ionization
enzyme linked immunosorbent assay	esi

Table 8.8: This table shows the most similar terms from the DTs for the term *mass spectrometry*.

Unified Medical Language System (UMLS)

The Unified Medical Language System (UMLS)[86] (Bodenreider, 2004) is an ontology for the medical domain. In contrast to the system by Szarvas et al. (2013), which uses WordNet (Miller, 1995) to generate substitute candidates and also for generating features, we use UMLS solely for feature generation.

Google Web1T

The Google Web1T Brants and Franz (2006)[87] is a resource of n-grams with counts, which was collected from text of publicly available websites. We use this resource as we expect this

[85] http://www.nlm.nih.gov/bsd/licensee/2014_stats/baseline_med_filecount.html

[86] available from http://www.nlm.nih.gov/research/umls/.

[87] http://catalog.ldc.upenn.edu/LDC2006T13

resource covers the open domain as well as most specific domains. For accessing the resource, we use JWeb1T[88] (Giuliano et al., 2007).

8.2.4 Lexical Substitution Dataset

The medical lexical substitution dataset was created by IBM T. J. Watson Research, Yorktown Heights, NY. The annotators were provided with a clear task: a question and a passage that contains the correct answer to the question were presented. This was restricted to a subset of passages that were previously annotated as justifying the answer to the question. This is related to a textual entailment task; essentially the passage entails the question with the answer substituted for the focus of the question. The annotators were then instructed to first identify the terms that were relevant for the entailment relation. For each relevant term, 10 terms from the ESG-based DT were randomly extracted within the top 100 most similar terms. Using this list of distributional similar terms, the annotators selected those terms that would preserve the entailment relation if substituted. This resulted in a dataset of 699 target terms with substitutes. On average from the 10 terms, 0.846 are annotated as correct substitutes. Thus, the remaining terms can be used as false substitute candidates.

The agreement on this task by Fleiss Kappa was 0.551 indicating "moderate agreement" (Landis and Koch, 1977). On the metric of pairwise agreement, as defined in the SemEval lexical substitution task, we achieve 0.627. This number is not directly comparable to the pairwise agreement score of 0.277 for the SemEval lexical substitution task (McCarthy and Navigli, 2009) since in our task the candidates are given. Nevertheless, it shows promise that subjectivity may be reduced by casting lexical substitution into a task of maintaining entailment.

8.2.5 Evaluation

For the evaluation we use a 10-fold CV and report on $P@1$ (Precision at one) and mean average precision (MAP) (Buckley and Voorhees, 2004) scores. The $P@1$ score indicates how often the first substitute of the system matches the gold standard. The MAP score is the mean of all AP from 1 to the number of all substitutes.

Features

We rely on four different information sources in order to create features, which are explained next:

- Google Web1T:
  We use the same Google n-gram features, as used in (Giuliano et al., 2007) and in (Szarvas et al., 2013). These are frequencies of n-grams formed by the substitute candidate s_i and

[88] https://github.com/dkpro/jweb1t

the left and right words, taken from the context sentence, normalized by the frequency of the same context n-gram with the target term t. Additionally, we add the same features, normalized by the frequency sum of all n-grams of the substitute candidates. Another feature is generated using the frequencies where target term t and the substitute cnadidate s_i are listed together using the words *and*, *or* and *","* as separator and add the left and right words of that phrase as context. Then, we normalize this frequency by the frequency of the context occurring only with t.

- DT features:
  To characterize if a target term t and a substitute candidate s_i have similar words in common, and thus are similar, we compute the percentage of words their thesauri entries share, considering the top n words in each entry with $n = \{1, 5, 20, 50, 100, 200\}$. During the DT computation we also calculate the significances between each word and its context features (see Section 8.2.3). Using this information, we compute if the words in the sentences also occur as context features for the substitute candidate. A third feature group relying on DTs is created by the overlapping context features for the top m entries of t and s_i with $m = \{1, 5, 20, 50, 100, 1000\}$, which are ranked regarding their significance score. Whereas the similarities between the trigram-based and the ESG-based DT are similar, the context features are different. Both feature types can be applied to the two DTs. Additionally, we extract the thesaurus entry for the target word t and generate a feature indicating whether the substitute s_i is within the top k entries with $k = \{1, 5, 10, 20, 100\}$ entries.[89]

- Part-of-speech n-grams:
  To identify the context of the word we use the POS tag (only the first letter) of s_i and t as feature and POS tag combinations of up to three neighboring words.

- UMLS:
  Using UMLS we look up all concept unique identifiers (CUIs) for s_i and t. The first two features are the number of CUIs for s_i and t. The next features compute the number of CUIs that s_i and t share, starting from the minimal to the maximum number of CUIs. Additionally, we use a feature indicating that s_i and t do not share any CUI.

Substitute candidates

The Substitute candidates for each target term are extracted from the ESG-based DT. For each target term, we use the gold substitute candidates as correct instances. Then we add all possible substitutes for the same target term as false instance, which occur in a different context, and are not annotated in the present context.

[89] Whereas in (Szarvas et al., 2013) only $k = 100$ is used, we gained an improvement in performance when also adding smaller values of k.

8.2.6 Results

Running the experiment, we receive the results as shown in Table 8.9. As baseline system, we use the ranking of the ESG-based DT. We already observe a quite high baseline, which can be attributed to the fact that this resource was used to generate substitutes and thus, contains all positive instances. Using the supervised approach, we can beat the baseline by 0.10 for the MAP score and by 0.176 for the P@1 score, which is a significant improvement ($p < 0.0001$, using a two tailed permutation test). To obtain insights of the contribution of individual fea-

System	MAP	P@1
Baseline	0.6408	0.5365
ALL	0.7048	0.6366
w/o DT	0.5798	0.4835
w/o UMLS	0.6618	0.5651
w/o n-grams	0.7009	0.6252
w/o POS	0.7027	0.6323

Table 8.9: Results for the evaluation of our lexical substitution system using substitute candidates from the DT. Additionally, we show results without specific feature groups.

ture types, we perform an ablation test. We observe that the most prominent features are coming from the two DTs as we only achieve results below the baseline when removing DT features. We still obtain significant improvements over the baseline when removing other feature groups. The second most important feature comes from the UMLS. Features coming from the Google n-grams improve the system only slightly. The lowest improvement is derived from the POS features. This leads us to summarize that a hybrid approach for feature generation using manually created resources (UMLS) and unsupervised features (DTs) leads to the best result for lexical substitution for the medical domain.

8.2.7 Analysis

For a better insight into the lexical substitution, we analyzed how often we outperform the baseline, receive equal results or obtain decreased scores. According to Table 8.10 in around

performance	# of instances	Avg. Δ MAP
decline	180	-0.16
equal	244	0
improvements	275	0.26

Table 8.10: This table shows the error analysis for the lexical substitution algorithm in comparison to the baseline based on the number of instances and respectively to the MAP scores.

26% of the cases, we observe a decreased MAP score, which is on average 0.16 smaller than the

scores achieved with the baseline. However, we also observe improvements with our method in around 39% of the cases. On average the improvements are 0.26, which is much higher than the loss. For the remaining 25% of cases the performance does not change.

Looking inside the data, the largest error class is caused by antonyms. A sub-class of this error class contains multiword expressions including adjective modifiers. Creating additional features using a lexical resource (e.g. UMLS), these problems might be solved. An example for an antonymy error is shown in Figure 8.3.

Sentence: The most common cause of thrombocytopenia during pregnancy is gestational thrombocytopenia, which is a mild thrombocytopenia with platelet levels remaining greater than 70,000/mL.

Gold: decreased platelet=1

DT: decreased platelet:17.0, severe thrombocytopenia:16.0, macrothrombocytopenia:16.0, prolonged aptt:16.0, normal platelet count:16.0, hepatosplenomegaly:15.0, hypoxaemia:13.0, short finger:12.0, lymphadenopathy:11.0, mild symptom:11.0

CT: severe thrombocytopenia:0.272, normal platelet count:0.204, macrothrombocytopenia:0.190, hepatosplenomegaly:0.174, prolonged aptt:0.168, decreased platelet:0.156, mild symptom:0.113, lymphadenopathy:0.085, hypoxaemia:0.067, short finger:0.053

Figure 8.3: Example sentence, where the target term *mild thrombocytopenia* is replaced. We show the replacement given by the annotator (Gold), the most similar terms extracted from a DT and the contextualized results from our system (CT). Here, our system returns a wrong ranking, as the adjective changes the meaning and turns the first ranked term into an antonym.

For feature generation, we look up multiword expressions as one term, in both the DT and the UMLS resource and do not split them into their single tokens. This error also suggests considering the single words inside the multiword expression, especially adjectives, and looking them up in a resource (e.g. UMLS) to detect synonymy and antonymy.

Figure 8.4 shows the case, where the ranking is performed correctly, but the precise substitute is not annotated as a correct one. The term *nail plate* might be even more precise in the context as the manually annotated term *nail bed*.

Due to the missing annotation, the baseline receives higher scores than the result from the system.

8.2.8 Conclusion

In this section, we have examined the lexical substitution task for the medical domain. We demonstrated how a system for open domain text data is transformed to the medical domain. For this, we replace the taxonomy WordNet with the UMLS ontology. Furthermore, we compute DTs based on medical text data and use a dependency parser, which is adjusted to the medical domain. We demonstrate that features generated with information from DTs have the

Sentence: 7 Yellow nail syndrome is a rare disorder characterized by the triad of yellow and thickened nails, lymphedema and respiratory manifestation commonly pleural effusion and other complications like bronchiectasis and chronic sinusitis. Lymphedema in yellow nail syndrome is characteristically non-pitting and involves the lower extremities in symmetric fashion

Gold: finger nail=1, nail bed=1

DT: finger nail:58, nail bed:54.0, nail plate:49.0, scalp:47.0, hand:26.0, arms and legs:25.0, eye:23.0, cranial bone:19.0, palm:16.0, urinary organ:16.0

CT: nail plate:0.676, finger nail:0.158, nail bed:0.144, scalp:0.106, hand:0.044, arms and legs:0.043, urinary organ:0.017, eye:0.016, palm:0.016, cranial bone:0.014

Figure 8.4: Example sentence for the target term *nails*. We provide the replacements given form the annotators (Gold) and the replacement from a DT and the ranked words form our lexical substitution system (CT). Here the ranking from the system is correct, but the first substitute from the system has not been annotated as correct replacement.

highest impact on the performance, followed by features from UMLS. Here, we have shown that the domain adaptation is possible and successful if we can replace resources by domain-adapted counterparts. Whereas the generation of unsupervised resources (e.g. DT) is feasible with less effort, this becomes more elaborate for manually built resources (e.g. UMLS).

8.3 Supersense Acquisition

> Words are pegs to hang ideas on
>
> Henry Ward Beecher, in *Proverbs from Plymouth Pulpit*

In WordNet (Miller, 1995) words referring to the same concept (e.g. synonyms) are grouped into so-called *synsets*. These synsets are again grouped into *lexicographer classes*. During the development process, 45 of these classes have been proposed to organize synsets of different POS. For instance, *body* and *animal* are lexicographer classes for nouns and *communication* and *emotion* are lexicographer classes for verbs. As these classes also represent senses on a coarse-grained level, Ciaramita and Johnson (2003) named them *supersenses*. In contrast to regular *term-based senses*, which are defined for each term individually, *supersenses* are categories that cover the complete vocabulary and are more coarse-grained.

In this chapter, we present an unsupervised method for the acquisition of supersense clusters, which uses a DT in conjunction with LDA (see Section 3.1.3). Using statistical semantics, we observe that for a term the most similar terms might refer to different senses. This is illustrated by the ten most similar terms for *drive* based on a DT in Table 8.11, which is computed based on Wikipedia and uses the *Stanford holing operation* (see Section 2.4).

term	supersense
disk	computer
touchdown	sport
pass	sport
punt	vehicle
disc	computer
controller	computer
card	computer
device	computer
motor	vehicle

Table 8.11: The ten most similar terms for *drive* extracted from an English DT computed on Wikipedia. Additionally, we manually labeled each term with an appropriate supersense.

According to the data we observe different meanings of the word *drive*: 1) the hardware device (e.g. harddisk drive), which is computer related, 2) the golf sport meaning, as *drive* is the first hit into the direction of the hole and 3) a vehicle related sense. The different senses of terms can be disambiguated per term using clustering algorithms, as shown in (Gliozzo et al., 2013) and (Riedl et al., 2014c). This method relies on a DT and uses Chinese Whispers (CW) (Biemann, 2006) to cluster the senses for each term based on the context features similar words share. Using this clustering method requires computing senses for each term separately. For this reason, we refer to these senses as *term-based senses*, as these clusters are not connected among other terms. In this section, we introduce a method to automatically create N supersense clusters, which cover the senses for all words.

8.3.1 Related Work

The work of Ciaramita and Johnson (2003) uses supersenses from WordNet to predict the supersense of a word in context in a supervised fashion. But in this work we apply an unsupervised method, which uses a Topic Model (TM). Using Topic Models (TMs) for computing senses instead of topics was performed by Preiss and Stevenson (2013) and Yao and Van Durme (2011). For this, most methods require sense-annotated corpora. However, sense annotated corpora need to be generated, either by manual annotation or by Word Sense Disambiguation (WSD) systems, which again need to be trained by manually created data. Alternatively, word sense annotated data can also be generated using an unsupervised WSD system (e.g. Agirre and Soroa (2009)). Guo and Diab (2011) presented a variation of LDA in order to detect senses by training the model on WordNet. Contrary to these methods, our method does not require sense annotated data but learns the supersense clusters based on the information given by a DT. Such clusters can be computed automatically without the need of manually created resources. Lau et al. (2012) introduce a Word Sense Induction (WSI) system, which uses LDA

and Hierarchical Dirichlet Process (HDP)[90] and does not need manually annotated data. In contrast to our method, an LDA model is computed for each word separately. This approach does not scale well when computing senses for all words.

The computation of LDA with different data than text has been performed by Ritter et al. (2010) and Melamud et al. (2013). In these works, arguments and predicates are used as *pseudo documents* to retrieve similarities among them. Andrews et al. (2009) introduces an LDA extension, which allows having a word and a context representation for each topic. This has shown to be beneficial for psycholinguistic tasks (Andrews et al., 2009) and word similarities (Roller and Schulte im Walde, 2013).

8.3.2 Method

To compute supersense clusters, we rely on a DT and LDA. In standard LDA, the model is computed based on documents and assumes that each document is described by a topic distribution and each topic is a distribution over words. Following our approach, we replace a document by a *pseudo document*, which is a list of similar terms, coming from a DT. Using similar terms as document, we expect this pseudo document not to follow a topical distribution, but a supersense distribution.

For the generation of the supersense clusters, we extract terms from a DT with a frequency above w. When POS tags are available in the DT, we could also restrict the supersense clusters to cover e.g. nouns only. Even though, a separation between different POS tags is achieved when mixing words of different POS within the pseudo document, we receive cleaner models when focusing on one POS and can use a lower number cluster for computing the LDA model. After the generation of the pseudo document, we run standard LDA and retrieve a model, which captures supersense categories.

These supersense clusters can serve as a feature for machine learning tasks and for supersense tagging (Attardi et al., 2010). Yet, these supersense clusters do not contain contextual information. Thus, we also introduce an extension to add context features to each topic. During the similarity computation (see Section 5.3), we obtain the most significant context features for each term. These can be added to the pseudo document, containing the similar terms. A more sophisticated approach is achieved by using the LDA extension by Andrews et al. (2009). In this extension, each topic is represented by a term topic and a context topic. Thus, a mixture of *language elements* and *context features* in the same topic space is avoided, which is more reasonable as distributions of *language elements* and *context features* might be different.

To demonstrate the impact of the method, we present the five supersense clusters with the highest probability for the noun *drive* in Table 8.12. Each cluster is represented by the most probable terms of each supersense cluster. For the example supersense clusters, we perform the computation solely for nouns and train LDA with 50 clusters and set $w = 10$. Whereas

[90] HDP is a topic model similar to LDA but the number of topics is estimated automatically.

Supersense 24		Supersense 3		Supersense 26		Supersense 4		Supersense 11	
software	0.0040	frame	0.0031	team	0.0027	car	0.0031	route	0.0049
system	0.0038	plate	0.0028	competition	0.0026	model	0.0021	road	0.0042
computer	0.0036	tube	0.0027	football	0.0024	motorcycle	0.0021	highway	0.0032
technology	0.0035	panel	0.0027	tournament	0.0023	engine	0.0021	line	0.0029
device	0.0035	pipe	0.0026	championship	0.0022	automobile	0.0019	lane	0.0029
application	0.0034	rod	0.0026	race	0.0021	vehicle	0.0019	railway	0.0026
hardware	0.0032	wire	0.0023	event	0.0021	prototype	0.0017	freeway	0.0025
interface	0.0030	door	0.0023	racing	0.0020	sedan	0.0016	railroad	0.0024
file	0.0028	wheel	0.0023	swimming	0.0020	bike	0.0015	Interstate	0.0023
datum	0.0027	lamp	0.0023	sport	0.0020	locomotive	0.0015	alignment	0.0022
..			...		...		...		
drive	0.0012	drive	0.0007	drive	0.0004	drive	0.0003	drive	0.0002

Table 8.12: List of terms for the supersense clusters computed with LDA, for which the term *drive* has the highest probabilities.

from the DT entry in Table 8.11 we have manually marked supersenses, here we show five supersense clusters. In the remaining 45 supersense clusters, the term *drive* has only low probability within the supersense cluster and they do not seem to be relevant for this term. The supersense cluster 24 describes drive in the sense of *hard disk drive* and supersense cluster 3 refers to drive as a *gateway*. Cluster 26 reflects the *sportive meaning* of the term drive, which refers to a long-distance shot in golf from the teeing ground. The terms in supersense cluster 4 are related to *vehicles*, as drive also refers to a journey by car. The remaining cluster 11 determines the supersense for *streets*, as drive can be also a part of a street name e.g. *Mulholland Drive*. We show the advantage of the supersense clusters when using them as features in an NER system as described in the following section.

8.3.3 Using Supersenses for Named Entity Recognition

In the work of Benikova et al. (2015)[91], a German Named Entity Recognition (NER) system[92] is introduced, which uses information from our supersense clusters and DTs as feature. The task of NER bothers with the classification and detection of named entities within text. These named entities are subdivided into categories like locations (e.g. New York, London), organizations (e.g. TU Darmstadt) or persons (e.g Luis Armstrong). The NER system is trained on the GermEval 2014 dataset called NoSta-D (Benikova et al., 2014)[93]. We give an example of a sentence from this dataset in Figure 8.5. We observe the three organizations (ORG), which are ice-hockey teams (*Eisbären*, *Sparta Prag* and *DEL-Kontrahent Adler Mannheim*). However, two

[91]Whereas the system has not been built by the author of this thesis, it uses features from supersenses and DTs that have been introduced in this thesis.

[92]The NER system is available at: `https://github.com/tudarmstadt-lt/GermaNER`.

[93]The dataset is freely available at: `https://sites.google.com/site/germeval2014ner/data`.

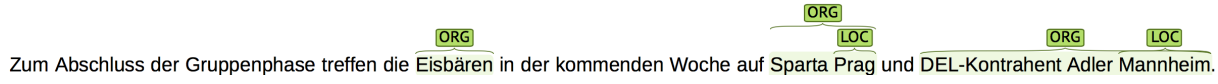


Figure 8.5: This sentence from the NoSta-D (Benikova et al., 2014) dataset contains annotated named entities. Sometimes annotations are nested, as for the ice-hockey organization (ORG) *Sparta Prag*, which also contains the location (LOC) *Prag*.

organizations also contain location name (LOC) annotations (*Prag* and *Mannheim*). Within the proposed NER system, only non-overlapping entities are used for learning and classification. The system uses Conditional Random Fields (CRF), a supervised classifier, and achieves almost state-of-the-art performance.[94]

Benikova et al. (2015) use character n-grams (suffix and prefix) features, word n-grams, unsupervised POS features, named entity lists from Freebase[95] and similarities from a DT and supersense clusters. All the features are extracted for the current token, the two preceding and two following features.

The DT is build using the method introduced in Chapter 5. We compute a DT based on 70 million sentences from German newspaper obtained from the Leipzig Corpora Collection (LCC). For the extraction of the term and context features, the trigram holing operation is used, that extracts for term the left and right neighboring word as context and is explained im more detail in Section 2.4. Then, we use the four most similar terms for each term from the DT as a bag-of-word feature.

The same DT is also used to compute the supersense clusters. We generate several clusterings based on all words within the DT with a frequency above five ($w = 5$). As the DT does not contain any POS information, the supersense clusters are computed for all POS tags at once, i.e. this setup is POS-agnostic. In the German language, nouns normatively start with an uppercase letter. Thus, we additionally compute clusters using only terms, which start with an uppercase letter, leading to supersense clusters, containing mainly nouns. Both supersense clusters are computed with 50, 200 and 500 senses. As features, we use the cluster identifier of the highest three senses per term. Experiments showed that using 200 supersense clusters results to the best performance.

Based on the development dataset an ablation test was performed. We present the results of the best feature groups, taken from Benikova et al. (2015), and present the performance of the features generated with information from DTs and supersense clusters in Table 8.13. According to the F1-measure, the character n-gram features have the most impact on the result. Considering the single feature groups, the DT features are the third best performing features and the supersense clusters are ranked at position nine. Combining the DT and supersense features as one feature group, they are the second most important feature group. Whereas the

[94] In comparison to the competitive systems, it uses features and resources, which are available following permissive license.

[95] `www.freebase.com`

Model	Precision	Recall	F1-measure
All features	83.16	74.32	78.49
no character n-grams	82.18	69.81	75.49
no case information	81.93	73.29	77.37
no DT	82.29	73.25	77.50
no supersense clusters	82.48	73.60	77.79
no DT and no supersense clusters	82.72	72.17	77.08

Table 8.13: Results for an ablation test based on the development set for the German NER system (Benikova et al., 2015).

DT feature seems to be more important for precision rather than recall, the reverse is shown for the supersense features. Both semantic resources deliver information to build features, which are worth to integrate as an improvement for the F1-measure of 1.41 is obtained.

System	Precision	Recall	F1-measure
Christian Hänig (2014)	82.72	71.19	79.08
Benikova et al. (2015)	80.67	77.55	76.52

Table 8.14: Results on the test set for the best performing system Christian Hänig (2014) and the system using our features introduced by Benikova et al. (2015)

Based on the test set (see Table 8.14), the NER system by Benikova et al. (2015) achieves higher precision than the best competing system (Christian Hänig, 2014) but has much lower recall.

Conclusion

In this section, we have introduced a method for creating supersense clusters. We show within an ablation test in an NER system proposed by Benikova et al. (2015) that using the cluster information as well as similarities from a DT improves the performance of this system. The supersense clustering method uses similarities from a DT as pseudo documents, which are disambiguated by LDA. In contrast to word-related sense clusters, which need to be computed for each word separately, we only need to compute the clusters once covering the supersenses for all words within the pseudo documents. A further modification allows integrating context features explicitly into the LDA model. Such a modification gives the possibility to apply the supersense clusters within a WSI system. Whereas we obtain noun supersense clusters when using only nouns within the pseudo documents, we observe a separation by their POS when considering all words. This might be suited for building a topic-based unsupervised POS tagger.

Chapter 9

Outlook

Угаанла төнчу чок.
Mind has no end

- Tuvanian adage

Within the previous chapters, we introduced methods for text segmentation, similarity computation, MWE detection and applications using these semantic resources and algorithms. In this chapter, we direct to further research directions, not covered within this thesis, which mainly apply information from DTs.

9.1 Tracing the Meaning of a Word

In Chapter 5, we introduced a symbolic method for computing similarities between *language elements*. Whereas the similarities rely on the corpus (cf. Section 5.7.5), which is used for the computations, it does not distinguish between different meanings of the *language element*. In Section 8.3, we introduced a method for detecting supersenses for *language elements* after computing similarities. In (Gliozzo et al., 2013; Riedl et al., 2014c) we have computed senses for each *language element* using a DT. Based on these senses, we have also examined the change of word senses over time (Mitra et al., 2014, 2015). Whereas these are methods, which figure out the senses based on its corpus, in some cases we already know the different meanings of *language elements* already before computing the similarities.

When we process newspaper data, we expect to have meanings of *language elements* coming from an open domain and a medical meaning is retrieved when performing the computations on medical data. We show the similarities for the term *cleavage* for both corpora in Table 9.1. In the open domain, a *cleavage* is the area between a woman's breasts. This can be also observed from the similar terms shown in the left column. The ten most similar terms are body parts and terms related to cleavage. In the medical domain the term *cleavage* is used in

Open domain	Medical domain
midriff#NN	proteolysis#NN
bosom#NN	Cleavage#NP
divide#NN	digestion#NN
rift#NN	degradation#NN
divides#NN	hydrolysis#NN
chest#NN	processing#NN
buttock#NN	cleavage#NP
abs#NN	fragmentation#NN
cheekbone#NN	scission#NN
breast#NN	activation#NN
neckline#NN	modification#NN
tension#NN	synthesis#NN

Table 9.1: This table shows the ten most similar terms including their POS for the noun *cleavage*. The similarities are computed based on the Stanford dependency parsing holing operation. We show similarities computed on the open domain (left column) and the medical domain (right column).

the field of mass spectrometry to describe the process where an enzyme is applied to fragment proteins into smaller fragments, called peptides. This process is also called *protein digestion*. We observe that the similar terms for *cleavage* in the right column are within the mass spectrometry domain, as we observe similarities to e.g. *proteolysis* (the breakdown of proteins into amino acids or polypeptides), *digestion*, *fragmentation* or *hydrolysis* (the cleavage of chemical bounds using water).

Whereas different senses within the same domain need further processing, the domain of the text that is processed is often known. In addition, time information is often available e.g. in newspaper text or books. Here, we describe a method, which uses this information in text to obtain the meaning of *language elements* according to this additional information.

Method

Word tracing is the name of the method for keeping track of the context of a *language element* it is extracted. The method relies on the similarity computation method described in Chapter 5 but uses a modified holing operation. Namely, we append additional information to the *language element* but do not change the *context features* for computing similarities. Adding context related information to a term has already been introduced with the *collapsed dependency parse holing operation* described in Section 2.4.2. This holing operation appends the POS tag to each term.

Appending the POS information is performed for each term separately. Here, we append not only term-relevant information but also information we extract from a complete document such as domains or the year it was published. For this approach, corpora are required,

which contain domain or time information. Such information is e.g. contained in the Google's syntactic n-gram dataset (Goldberg and Orwant, 2013), which contains syntactic dependencies including frequencies for each year. Appending the year to each term might result into sparse representations. Thus, using time spans is preferred. Considering the example given in the introduction we can use the corpus information and add the source to the terms resulting into terms like *newspaper#cleavage* and *medline#cleavage*. Additionally, the term itself without any tracing should be included, to still capture the most prominent meaning of a term. Using this term representation, we observe different terms to be similar to the corpus-marked terms. Such information is useful for different applications: Using corpus information e.g. adding the domain, gives results over different meanings of words within its domain. The usage of time information allows detecting changes of the meaning for a term over time and might allow rephrasing old-fashioned terms by newer ones. Similar research has been proposed by Rapp (2004), who computes senses using several corpora. In contrast to our approach, he uses co-occurrences of words and extracts two main senses per term.

In this section, we have described a method for tracing back the meaning of a word regarding to its appearance. This is not only restricted to domain or time-span detection but can be applied to further applications (e.g. sentiment annotation, author annotations etc.). When extending the approach to n-grams, we might be able to figure out related phrases for different domains. For example *catching a cold* might be similar to *lose money* in the finance domain, as they might share the same context representation.

9.2 Automatic Computation of Domain-specific Thesauri

Using *word tracing* introduced in the previous section, we can build domain-aware thesauri. But the approach needs to know the domains, which are appended to the terms beforehand. Here we describe an extension of the tracing method where it can be used to build a domain specific DT unsupervised and knowledge-free. Additionally, we introduce a method to obtain a DT, which is suited to the data it is applied.

Method

First, we compute a topic model based on LDA using a corpus, which covers different domains. Thus, an encyclopedia (e.g. Wikipedia) is a good choice as it covers a wide range of different domains and topics. After the training of the topic model, we achieve n topics, which are represented by words with probabilities belonging to the topic. Additionally, each word in the corpus used for training is assigned with a topic identifier. In the second step, we apply the *word tracing* mechanism by appending these topic identifiers to the terms and then compute similarities between this extended terms. In addition, we add the term without any topical information. This results to a DT with similarities for different topics.

If we require a DT, which represents terms from e.g. topic 3 and 101 we extract *language elements* having the trace of the selected topic. Whereas this process still needs manual selection, we present another options, which automatically selects the topics that need to be extracted. In most NLP applications, we have textual data, which represents the domain the task is applied. Often this data is too small to result to DTs with high vocabulary coverage. But we can use this textual data and apply the trained topic model to estimate the topic distributions of the document. This topic distribution can then be used as a selector for extracting relevant DTs, which can then be combined.

As an alternative, we can produce DTs for each topic. For this, we use for each topic-related DT only documents, which have the highest probability for this topic. If documents are mixed with different topical context, we can apply TopicTiling (see Chapter 4) in order to achieve topically coherent text segments. The selection of the fractions of the domain DTs is decided based on the topic distribution of a previously unseen document. In most cases, we have a DT of much higher quality for the open-domain, as there might be more data available than for special-domains. If this is the case, an open-domain DT can be used as a back-off model for unknown terms in the domain-based DT.

In this section, we have illustrated how we can build a domain-specific DT automatically with a high coverage of words, even when the domain-specific corpora are rather small. Whereas we have described two possible alternatives for building domain specific DTs, we also have illustrated how domain-related and open-domain DTs can be combined.

9.3 Populating Ontologies

Ontologies are resources describing entities (e.g. *Pulp Fiction*) by its concepts (e.g. *movie*) and properties (e.g. *actors*, *director*, etc.). Most ontologies are built manually (e.g. Baker et al., 2006; Bodenreider, 2004) or semi-automatically using linked encyclopedias like Wikipedia (e.g. Lehmann et al., 2015; Suchanek et al., 2007). As most resources are incomplete, we describe a method for introducing new entities into an existing ontology using a DT. This is different from entity linking (Rao et al., 2013), which is the task of detecting entities within text and linked them to the matching entity within an existing ontology. As a restriction, entity-linking systems can only detect entities, which are contained in the knowledge base. Here we describe a method, which adds new entity candidates into existing ontologies by using its context representation.

Method

Ontologies contain entities, which are linked to concepts. This is similar to the WordNet taxonomy, which distinguishes between types (concepts in ontologies) that are common nouns

and instances (entities in ontologies) like specific persons or movies. In WordNet instances are connected to types relying on an *instance-of* relation.

To introduce new entities into an ontology, new entities need to be found. For detecting entities in text, a NER system (e.g. Nadeau and Sekine (2007); Benikova et al. (2015)) can be used. These systems are not restricted to detect entities, which are within a knowledge base but can also find new entities.[96] We use the entities found by the NER as *language element* and build a DT. Computing a DT does not only result to similarities between terms, but also we obtain context features for *language elements*. For the movie *Pulp Fiction* we find e.g. the context representations shown in Table 9.2. This context representation is generated using a holing operation, which extracts all co-occurring terms within a sentence. Using only neighboring words, might result into a sparse context representation, which is beneficial for computing similarities, but not for aligning entities to concepts.

Context	LMI score
Tarantino's#NP	291.88
Tarantino#NP	121.07
Thurman#NP	79.49
Miramax#NP	65.50
Heiser#NP	58.79
Travolta's#NP	56.07
Winnfield#NP	54.93
Lofficier's#NP	46.02
Joffé#NP	39.68
Travolta#NP	39.52
Misirlou#NP	38.41
Reece's#NP	37.85
Jean-Marc#NP	34.21
Heidenreich#NP	33.64
Dale's#NP	32.59
escalator#NN	32.46
McMahon's#NP	31.20
Travolta#NP	30.57
Uma#NP	30.15

Table 9.2: This tables shows the highest scored co-occurring contexts including POS tags for the entity *Pulp Fiction*.

These *context features* serve as a representation for the entity. Exemplary, we assume that the entity *Pulp Fiction* is not contained in the ontology and the goal is to add it into the correct concept. First, we build a context representation for each concept. Then, we extract all entities for all concepts. For the movie concept, we might find further entities e.g. *Reservoir*

[96] As further extension we can also use the MWE detection described in Chapter 7 to find entities.

Dog, *Fargo*, *Be Cool*. We extract the context representation for all entities of a concept and merge the representations to form a general concept representation. This representation is then generated for all concepts. For assigning the entity *Pulp Fiction* into the ontology, we can use the cosine similarity to find the most similar concept for the entity.

To evaluate the method, one can generate training data by extracting concepts and its entities from an existing ontology. For this, we remove some entities and try to assign them back to the correct concept. This allows building a dataset automatically and provides the possibility to evaluate of the method.

We have described a method for enriching ontologies by adding new entities. Furthermore, we have laid down an automatic method to create a dataset to test the described method. This is a further step in applying NLP techniques in order to improve knowledge bases used in the field of Semantic Web.

Chapter 10

Conclusion

So Long, and Thanks for All the Fish

Douglas Adams - in *The Hitchhiker's Guide to the Galaxy*

In this thesis we presented methods, which take computers one step further to understand language by processing text. This has been achieved by using statistical semantics in order to extract new structure from text. As structure we refer to grammatical, semantic or syntactical structure. Such structure can be used for further structure discovery steps and serves as relevant information in order to understand natural language. To extract structure from text we proposed methods, which perform the extraction from arbitrarily large data using unsupervised methods and focusing on language-independent processing.

In this chapter, we summarize the findings of this thesis and highlight the contributions. Furthermore, we describe the limitations and open issues of the methods proposed. Then we exemplify our methods, by showing the structure they can induce into text. Afterwards, we conclude this thesis with final remarks.

10.1 Summary and Contributions

Humans retrieve the meaning (semantics) of a word by the context it appears (e.g. the sentence, paragraph, neighboring words). Computers can process text and extract contexts (e.g. neighboring words) for all words. This results to a semantic representation, which can be used to compute semantic similarities between words.

Background

In Chapter 2 we introduced the terminology used in this thesis. Furthermore, we gave a brief description for processing methods in NLP. In the following section we described the linguistic theory that our methods are based on. Then, we introduced a graph-based formalism, which was used to compute similarities based on a context representation. This context representation has been instantiated afterwards, using the graph-based formalism. This provides a generic approach that separates between language elements (e.g. words, n-grams, sentences) and context features (e.g. neighboring words).

Semantic Similarity

In Chapter 3 we described existing methods, which have been designed to compute semantic similarities between words and documents, and illustrated these with a running example. Depending on the method and the context used for the similarity computation, similarities vary from topical relatedness between words up to near-synonymy. Furthermore, we divided the algorithms in symbolic and non-symbolic approaches. Whereas symbolic approaches allow to provide reasoning for the similarity of two terms by representing the shared context in a human-readable fashion, non-symbolic approaches rely on a dense numeric vector representation. Although, the similarities of non-symbolic approaches are in line with the ones of symbolic approaches, the interpretation of dense vectors is hardly possible.

Topically Text Segmentation

In order to understand language, humans segment text into units. Whereas tokenization of words is a solved problem for computers, considering languages with space-separated words, the segmentation of text into topically coherent segments is still a challenging task. In Chapter 3 we have shown that, for the task of text segmentation, topic-based word representations outperform word-based models. This was demonstrated for the text segmentation algorithms C99 and TextTiling by replacing words with its contextualized topic identifiers. As these algorithms are designed to perform the text segmentation using words, we implemented an algorithm called TopicTiling. TopicTiling is a simplified and topic-based adaptation of TextTiling and achieved state-of-the-art performance on two representative datasets. Additionally, we have shown the influence of the parameters for the topic model LDA based on the performance of the text segmentation task. This constitutes the first detailed extrinsic evaluation for the parameters of LDA.

Symbolic Similarity Computation

In Chapter 5 we introduced a generic framework for computing similarities between language elements (e.g. tokens, words or sentences), leading to a distributional thesaurus (DT). Our approach uses a symbolic context representation and scales to arbitrarily large amounts of data. Furthermore, it is not limited to compute similarities between terms using specific context representations, but allows computing similarities between different kinds of language elements and supports the usage of various context representations.

We demonstrated the influence of different parameter settings and showed the impact of different context feature representations. The evaluation for the similarities was performed for 2000 nouns against the lexical database WordNet. The best performance was achieved by using syntactic dependency parses for the context representation of terms. Furthermore, we have shown that the quality of a DT increases with the amount of text used for the computation. However, if no preprocessing tools like syntactic dependency parsing is available for a language or a domain, we advise to use as much data as available and apply as context representation the left and right neighboring word (trigram holing operation).

We demonstrated, that our method outperformed two standard algorithms for similarity computation when scaling to large amounts of data. In addition, our method outperforms the currently popular method called *word embeddings* when computing these embeddings with the default parameters of `word2vec`. However, we observed a better performance with word embeddings that use tuned parameters. Additionally, we detected that lower ranked similar words extracted with word embeddings are inferior to the ones computed with our method. In an comparison of word embeddings using syntactic dependencies with or method, we observe that our method outperforms these embedding representation and in an evaluation these adapted embeddings score only slightly higher compared to the standard word embeddings. In contrast to both word embedding methods, our method follows a symbolic context representation that is able to provide reasoning for the similarities, which is e.g. important for sensitive applications in the medical or finance domain.

Evaluating the influence of different corpora used for the DT computation, we observed that using a small corpus of high quality yields higher evaluation scores than using a larger corpus of lower quality. In this experiment we have demonstrated that using huge corpora together with linguistic preprocessing performs best.

In a last experiment, we have compared the similarities of verbs against WordNet. Here we showed that our method for DT computation achieves the best overall performance. However, considering low frequent verbs, our methods score lower than the currently best system. In contrast to that system, we do not perform any parameter optimization for a specific POS and for terms of specific frequency spans.

Extrinsic Evaluation of Unsupervised Parsers

In Chapter 5, we demonstrated that using a supervised syntactic dependency parser for generating the context representation for a word results in DTs, which achieve the highest performance in our evaluation. As we targeted on the development of language-independent methods, we have examined the replacement of language-dependent parsers with unsupervised language-independent parsers in Chapter 6. We evaluated the performance of supervised and unsupervised dependency parsers extrinsically — using them as context representation for building DTs — with the WordNet Path measure. In this evaluation we observed that an unsupervised parser outperforms a supervised parser when used as context representation for building DTs on German text. Based on English language, unsupervised parsers cannot compete against the supervised parsers. As English is the most-studied language for supervised dependency parsers, it seems obvious that a supervised parser performs best. Interestingly, our extrinsic evaluation captured different aspects of parsers, since the best performing unsupervised parser based on an intrinsic evaluation does not perform best in our extrinsic evaluation setting. Another insight is the fact that our framework is capable to find the best context representation when combining different contexts. With this contribution we demonstrated the best practice on how to compute DTs of high quality. This provides an extrinsic evaluation framework for unsupervised parsers. Additionally, this is the first comprehensive extrinsic evaluation for unsupervised parsers.

Multiword Expression Extraction

A method for the extraction of unit-forming term sequences (e.g. *New York*), called multiword expression (MWE), was presented in Chapter 7. This method ranks term sequences according to their "multiwordness". Most existing ranking methods require language knowledge. Here, we introduced a method, which performs the ranking without language-dependent processing. Our ranking measure, called `DRUID`, relies on a DT that considers similarities between n-grams. In an evaluation, we showed that ranking n-grams with the `DRUID` measure yields the best performance in comparison to other ranking methods, although it does not require language-dependent processing. The evaluation was performed on an English medical dataset and on a French newspaper dataset.

Applying Methods in NLP systems

We demonstrated the impact of our semantic models, by applying information from DTs into three NLP tasks as described in Chapter 8. First, we introduced a general approach to tackle the out-of-vocabulary (OOV) problem. This problem deals with words, called OOV words, which did not occur in the texts that have been provided for the training of a supervised method. Thus, supervised methods often fail when predicting structure for these OOV words.

We demonstrated the impact of our method based on an existing POS tagger. For this, we replaced words, absent from the training data, with similar and known words extracted from our DT. Following this approach, we were able to reduce the error rate for the OOV words and improve the overall performance.

Our DTs already have shown impact for the task of lexical substitution for the open domain in previous work. In the second task we adapted this open domain system to the medical domain and computed DTs for the medical domain. Performing an ablation test, we discovered that features derived from a DT have the highest impact followed by the contribution achieved with lexical resources.

In the last section, we introduced a method, which detects the different meanings of a word. This method assigns words into coarse-grained clusters, called supersenses. For the computation, we rely on the topic model LDA and a DT. We demonstrated the impact of these supersenses and DTs by generating features of both. Afterwards we plugged these features into a supervised system for the recognition of named entities (e.g. companies, locations or names). Such a system detects entities like companies, names or locations in text. In an evaluation of this system, we observed a performance gain using DT-based features and supersense features.

Further Extensions

In addition to the previously described contributions, we have also proposed concepts for further methods, which rely on DTs. First we described a method for computing DTs, which incorporates available information (e.g. date of a text, domain of text) into the generation process. An additional extension is suited to automatically construct a DT for a specific domain, which can be determined automatically given text of this domain. The last concept suggested an approach, which allows to add new information into existing knowledge bases (e.g. ontologies or taxonomies).

10.2 Limitations and Open Issues

In this thesis, we have presented various methods, which achieve state-of-the-art performance. However, these methods also have limitations, which we describe in this section.

Semantic Similarity

In Chapter 3 we described various methods and showed their output, but we did not report on evaluation results. Whereas this would clarify which method performs best, we do not think that such information gives additional benefit. Although such scores serve as indicator for the quality of a resource, it is also important to examine the similarity output, which gives

further insight into these methods. However, the most meaningful results would be given by evaluating the methods extrinsically, by comparing their performance in further tasks.

Topically Text Segmentation

Detecting topical changes in text is achieved with our text segmentation algorithm TopicTiling. We have shown that our method performs well on two datasets. But both datasets are artificially generated, merging different texts. Additionally, the Choi dataset cannot be split in training and test documents, as there is a high overlap between documents. Furthermore, in most "naturally occurring" textual data, text of different topics is often already separated. A more challenging and realistic task would be the detection of e.g. section or paragraph segments in text.

Symbolic Similarity Computation

Computing similarities based on a symbolic representation (see Chapter 5) enables to the retrieval of similarities only to *language element*, which at least share one *context feature*. Using a dense numeric vector representation e.g. word embeddings or LDA, allows the similarity computation between all words. Furthermore, our similarity score, which is the overlap of significant *context features* two *language elements* share, is not a strict metric but proportional to a metric, as shown in Section 5.3. Whereas this might be a drawback of our method, there is yet no proof that language itself fits into a metric space.

Multiword Expression Extraction

In Chapter 7 we introduced a method for ranking n-grams according to their multiwordness. Whereas the approach is able to generate a list of MWEs, it is not capable to recognize MWEs directly in text. This should be targeted with a sequence-tagging algorithm e.g. CRF. Furthermore, if no POS filter is applied for pre-filtering candidates, we also rank some n-grams to the top, which contain stopwords. Whereas in this thesis we concentrate on language-independent methods, using a stopword list solves this issue.

Applying Methods in NLP systems

Whereas we have shown the performance of our method, a further evaluation of the other methods, e.g. Lin's DT or word embeddings, has not been performed. This would answer the question, which statistical semantic model performs best for a specific task.

General Limitations

We presented several unsupervised methods in this thesis, which extract structure without language-knowledge. In this thesis we have applied them solely to Western languages and in most cases only to texts of the English language. Whereas this was specified in advance, our methods might also operate well on other languages. However, for some languages, e.g. Chinese, where tokenization is not as simple, further language-specific modifications need to be performed first.

10.3 Enriching Text with our Methods

Before we conclude this thesis, we demonstrate how the methods implemented in this thesis provide structure in order to obtain a better understanding of text. First, we apply the unsupervised methods on the text of the entire English Wikipedia, without considering any annotations contained in Wikipedia. To give an expression of the performance, we concatenated excerpts from two Wikipedia articles and annotated the text with our methods as shown in Figure 10.1. The first four sentences are extracted from the *heavy metal* article and the five remaining sentences are taken from the *beer* article.

First we apply the text segmentation algorithm TopicTiling from Chapter 4. We show similarity scores among the neighboring sentences on the left side of the figure. The lowest similarity score that indicates a segment is detected between the fourth and fifth sentence. Whereas for humans this is a trivial task, it is complex for computers. Extracting such structures is beneficial for further tasks like text summarization.

Additionally, we extracted multiword expressions (MWEs) using the DRUID method that was introduced in Chapter 7. We observe overlapping MWEs like *heavy metal* and *metal bands* in the first sentence as well as band names like *Led Zeppelin* or *Black Sabbath*. In addition, phrases are detected as MWE units, like *increasing emphasis on*. Whereas not all marked units seem to be correct (e.g. *also used*, *fermentation of*), they still might be beneficial for further NLP tasks, such as relation extraction.

Additionally, we demonstrate the similarities for two terms based on two different DTs. First, we show similarities for the term *genre* using syntactic dependencies (Stanford dependency holing operation) to extract the context representation. Whereas we achieve similar terms like *subgenre*, we also observe specific genres from both music ("*electronical*") and movies ("*fiction*").

The different supersenses for the term *genre* are shown below for the four highest matching supersenses computed with the method introduced in Section 8.3. Supersense 12 denotes the music genre and supersense 30 illustrates the description of a genre itself, as we obtain terms like *behaviour* or *attitude*. The book genre is covered by supersense 56 and the movie genre by supersense 84.

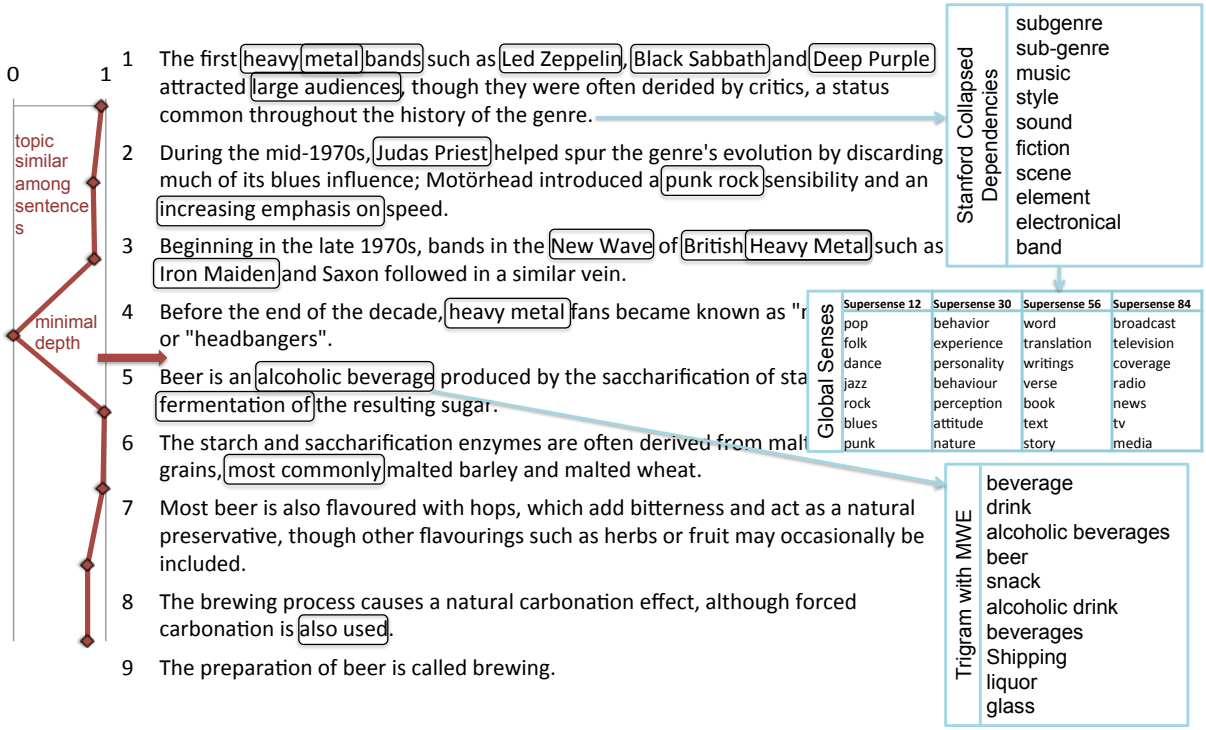


Figure 10.1: This figure shows two paragraphs from the Wikipedia article about *Heavy Metal* and *Beer*. On the left we show the cosine similarity scores between the different sentences. According to TopicTiling a segment, is found when a minimal depth is detected. This is the case between sentence 4 and 5. The black frames around words marks MWE detected by DRUID. For the terms *genre* and *alcoholic beverage* we show expansions from DTs using two different holing operations. Additionally, we show supersenses for the term *genre*.

Computing similarities, relying solely on neighboring words, results to terms, which do not only contain near-synonyms but also related terms. This is shown for the most similar terms to *alcoholic beverage*, which uses the trigram holing operation that considers the left and right word as context feature and uses n-grams as language element. Whereas most similar terms are beverages such as *liquor* and *beer*, we also obtain grocery-related terms e.g. *snack*.

10.4 Final Remarks

Within this thesis, we have shown that using statistical semantics and large amounts of textual data enables our methods to extract structure from text. This structure takes computers one step further to understand language. Our methods scale to arbitrarily large amounts of data and are also able to extract structure in a knowledge-free and unsupervised fashion. This is a major step in understanding not only a single language but “Language” in general.

Appendix

A Smoothed Cosine Similarity

The cosine similarity is the standard similarity measure in NLP and is often used for computing word similarities e.g. in VSMs (see Chapter 3). However, the cosine similarity performs poor when it is applied on sparse vector representations. Whereas there are previous approaches that integrate weightings into the cosine similarity e.g. Li and Han (2013), we introduce a simpler solution. First, we exemplify the sparsity issue using the following vectors:

$$X = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 100 \end{bmatrix}, \quad Y_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 150 \end{bmatrix}, \quad Y_2 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0.8 \end{bmatrix}, \quad Y_3 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 110 \end{bmatrix}.$$

If we compute the cosine similarity (see Equation 1) between X and $Y1$, $Y2$, $Y3$ we observe the highest similarity scores for $Y1$ and $Y2$ as shown in the third column in Table 1.

$$cosine\_similarity(X, Y) = \frac{\sum_{i=1}^{|X|} x_i \cdot y_i}{\sqrt{(\sum_{i=1}^{|X|} x_i^2) \cdot (\sum_{i=1}^{|Y|} y_i^2)}} \tag{1}$$

vector 1	vector 2	cosine	smoothed cosine		
			$\beta = 0.1$	$\beta = 1$	$\beta = 100$
X	Y_1	1.00000	0.99999	0.99999	0.98058
X	Y_2	1.00000	0.99240	0.63247	0.71274
X	Y_3	0.99996	0.99996	0.99995	0.99884

Table 1: Results for the similarities between X and Y_i using the cosine similarity and the smoothed cosine similarity with different β values.

However, comparing the vectors we might assume that Y_3 is more similar to X than Y_2. As Y_2 is parallel to X, the angle between the two vectors is zero and results to a the cosine similarity of 1.0. Whereas this is geometrically expected, we might prefer a similarity measure, that also reports lower score for parallel vectors if the components are not similar. Thus, we introduce an additional non-zero dimension, which can be seen as some kind of smoothing for the cosine similarity. Such an additional dimension avoids the vectors to be parallel. We introduce this modification as follows:

$$smooth\_cosine\_similarity(X, Y) = \frac{X\dot{Y}}{\sqrt{X^2 Y^2}} = \frac{\sum_{i=1}^{|X|} x_i \cdot y_i + \beta}{\sqrt{((\sum_{i=1}^{|X|} x_i^2) + \beta) \cdot ((\sum_{i=1}^{|Y|} y_i^2) + \beta)}}. \tag{2}$$

Using this new dimensions, we need to specify its constant value β. In Table 1 we report results for three different values of β. Considering a small smoothing value ($\beta = 0.1$), Y_1 is still most similar vector to X but is directly followed by Y_3. Using this parameter Y_2 is ranked lowest. With $\beta = 1.0$ we even obtain a similarity score of only 0.63 between X and X_2. Interestingly, with $\beta = 100$ we observe a completely different ranking. Here, Y_3 has the highest similarity score followed by Y_1 and Y_2. Although we have shown that smoothing sparse vectors might alleviate some issues, its impact needs to be evaluated in further research.

B Publications of the Author

Several of the main contributions of this thesis have been published in peer-reviewed conferences, workshops and journals of major events in natural language processing such as ACL, NAACL, EMNLP, COLING and LREC. The chapters that are build upon these publications are marked accordingly.

Peer-reviewed Journals

Sunny Mitra, Ritwik Mitra, Suman Kalyan Maity, **Martin Riedl**, Chris Biemann, Payal Goyal, Animesh Mukherjee, (2015): An automatic approach to identify word sense changes in text media across timescale, Natural Language Engineering, NLE, Special issue on Graph Methods for NLP, Vol. 21, pp. 773–798.

Chris Biemann, **Martin Riedl** (2013): 'Text: Now in 2D! A Framework for Lexical Expansion with Contextual Similarity', Journal of Language Modelling, JLM, 1(1), pp. 55–95 (Chapter 5 and Section 8.1)

Martin Riedl, Chris Biemann (2012): 'Text Segmentation with Topic Models', Journal for Language Technology and Computational Linguistics, JLCL, Vol. 27, No. 1, pp. 47–70 (Chapter 4)

Peter Findeisen, Diamandula Sismanidis, **Martin Riedl**, Victor Costina, Michael Neumaier (2005): Preanalytical impact of sample handling on proteome profiling experiments with matrix-assisted laser desorption/ionization time-of-flight mass spectrometry. Clinical Chemistry, Vol. 51, No. 12, pp. 2409–2411

Peer-reviewed Conference Proceedings

Martin Riedl, Chris Biemann (2015): 'A Single Word is not Enough: Ranking Multiword Expressions Using Distributional Semantics', in: Proceedings of the 2015 Conference on Em-

pirical Methods in Natural Language Processing, EMNLP 2015, pp. 2430–2440, Lisboa, Portugal (Chapter 7).

Tim Feuerbach, **Martin Riedl**, Chris Biemann (2015): 'Distributional Semantics for Resolving Bridging Mentions'. in: Proceedings of the Conference on Recent Advances in Natural Language Processing, RANLP 2015, pp. 192–199, Hissar, Bulgaria

Eugen Ruppert, Manuel Kaufmann, **Martin Riedl**, Chris Biemann (2015): 'JoBimViz: A Web-based Visualization for Graph-based Distributional Semantic Models', in: Proceedings of the Annual Meeting of the Association for Computational Linguistics, ACL 2015, System Demonstrations, pp. 103–108, Beijing, China

Eugen Ruppert, Jonas Klesy, **Martin Riedl**, Chris Biemann (2015): 'Rule-based Dependency Parse Collapsing and Propagation for German and English', in: Proceedings of the International Conference of the German Society for Computational Linguistics and Language Technology, GSCL 2015, pp. 58–66, Duisburg, Germany

Martin Riedl, Michael Glass, Alfio Gliozzo (2014): 'Lexical Substitution for the Medical Domain', in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, pp. 610–614, Doha, Qatar (Section 8.2)

Martin Riedl, Irina Alles, Chris Biemann (2014): 'Combining Supervised and Unsupervised Parsing for Distributional Similarity', in: Proceedings of the 25th International Conference on Computational Linguistics, COLING 2014, pp. 1435–1446, Dublin, Ireland (Chapter 6)

Martin Riedl, Richard Steuer, Chris Biemann (2014): 'Distributed Distributional Similarities of Google Books over Centuries', in: Proceedings of the Fourth International Conference on Language Resources and Evaluation, LREC 2014, pp. 1401–1405, Reykjavik, Iceland (Chapter 5)

Martin Riedl, Chris Biemann (2013): 'Scaling to Large^3 Data: An efficient and effective method to compute Distributional Thesauri', in: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, pp. 884–890, Seattle, WA, USA (Chapter 5)

Martin Riedl, Chris Biemann (2012): 'How Text Segmentation Algorithms Gain from Topic Models', in: Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2012, pp. 553–557 , Montreal, Canada (Chapter 4)

Peer-reviewed Workshop Proceedings

Alfio Gliozzo, Chris Biemann, **Martin Riedl**, Bonaventura Coppola, Michael R. Glass, Matthew Hatem (2013): 'JoBimText Visualizer: A Graph-based Approach to Contextualizing Distributional Similarity. In Proceedings of the 8th Workshop on TextGraphs in conjunction with EMNLP 2013, pp. 6–10, Seattle, WA, USA

Chris Biemann, **Martin Riedl** (2013): 'From Global to Local Similarities: A Graph-Based Contextualization Method using Distributional Thesauri', in: Proceedings of the 8th Workshop on TextGraphs in conjunction with EMNLP 2013, pp. 39–43, Seattle, WA, USA

Janneke Rauscher, Leonhard Swiezinski, **Martin Riedl**, Chris Biemann (2013): 'Exploring Cities in Crime: Significant Concordance and Co-occurrence in Quantitative Literary Analysis', in: Proceedings of the Computational Linguistics for Literature Workshop at NAACL-HLT 2013, pp. 61–71, Atlanta, GA, USA

Martin Riedl, Chris Biemann (2012): 'TopicTiling: A Text Segmentation Algorithm based on LDA', in: Proceedings of the Student Research Workshop of the 50th Meeting of the Association for Computational Linguistics, ACL 2012, pp. 37–42, Jeju, Republic of Korea (Chapter 4)

Martin Riedl, Chris Biemann (2012): 'Sweeping through the Topic Space: Bad luck? Roll again!', in: Proceedings of the Joint Workshop on Unsupervised and Semi-Supervised Learning in NLP held in conjunction with EACL 2012, ROBUS-UNSUP 2012, pp. 19–27, Avignon, France (Chapter 4)

Holger Storf, **Martin Riedl**, Martin Becker (2009): 'Rule-based activity recognition framework: Challenges, technique and learning', in: Proceedings of the 3rd International Conference on Pervasive Computing Technologies for Healthcare at PervasiveHealth, pp. 1–7, London, England

Ehrenwörtliche Erklärung[97]

Hiermit erkläre ich, die vorgelegte Arbeit zur Erlangung des akademischen Grades „Doctor rerum naturalium“ mit dem Titel *„Unsupervised Methods for Learning and Using Semantics of Natural Language“* selbständig und ausschließlich unter Verwendung der angegebenen Hilfsmittel erstellt zu haben. Ich habe bisher noch keinen Promotionsversuch unternommen.

Darmstadt, den 11. Januar 2016 (Martin Riedl)

[97] Gemäß § 9 Abs. 1 der Promotionsordnung der Technischen Universität Darmstadt.

Wissenschaftlicher Werdegang des Verfassers[98]

10/2005-02/2009	Diplom-Informatik Studium an der Hochschule Mannheim
09/2008-02/2009	Abschluss als Diplom-Informatiker (FH) Diplomarbeit: “Usage of data mining to learn activity recognition rules” am Fraunhofer IESE, Kaiserslautern
03/2009-03/2010	Informatik Master Studium an der Hochschule Mannheim
10/2009-03/2010	Abschluss als Master of Science Masterarbeit: “Using protein identification data to improve mass spectrometry feature extraction” am Heidelberg Collaboratory for Image Processing (HCI) an der Uni Heidelberg
04/2010-05/2011	Wissenschaftlicher Mitarbeiter am Universitätsklinikum Mannheim im Bereich der DNA Analyse
08/2013 - 10/2013	Praktikum bei IBM T. J. Watson Research, Yorktown Heights, NY, USA Entwicklung eines Wort-Substitutionssystems in der medizinischen Domäne
seit 06/2011	Wissenschaftlicher Mitarbeiter am Fachgebiet Language Technology an der Technischen Universität Darmstadt

[98] Gemäß §20 Abs. 3 der Promotionsordnung der Technischen Universität Darmstadt

List of Tables

List of Figures

Bibliography

Anne Abeillé and Nicolas Barrier: 'Enriching a French Treebank', in: *Proceedings of the Fourth International Conference on Language Resources and Evaluation*, LREC 2004, pp. 2233–2236, Lisbon, Portugal, 2004, Online: http://aclweb.org/anthology/L04-1347.

George W. Adamson and Jillian Boreham: 'The Use of an Association Measure Based on Character Structure to Identify Semantically Related Pairs of Words and Document Titles', *Information Storage and Retrieval* 10 (7-8): 253–260, 1974, Online: http://www.sciencedirect.com/science/article/pii/0020027174900205.

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa: 'A study on similarity and relatedness using distributional and WordNet-based approaches', in: *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL 2009, pp. 19–27, Boulder, Colorado, USA, 2009, Online: http://www.aclweb.org/anthology/N09-1003.pdf.

Eneko Agirre and Aitor Soroa: 'Personalizing PageRank for Word Sense Disambiguation', in: *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, EACL 2009, pp. 33–41, Athens, Greece, March 2009, Online: http://www.aclweb.org/anthology/E09-1005.

Dimitra Anastasiou: *Idiom Treatment Experiments in Machine Translation*, Ph.D. thesis, Universität des Saarlandes, Saarbrücken, Germany, 2010, Online: http://d-nb.info/1007560460/34.

Mark Andrews, Gabriella Vigliocco, and David Vinson: 'Integrating experiential and distributional data to learn semantic representations', *Psychological Review* 116 (3): 463–498, 2009, Online: http://www.ncbi.nlm.nih.gov/pubmed/19618982.

Roxana Angheluta, Rik De Busser, and Marie-Francine Moens: 'The Use of Topic Segmentation for Automatic Summarization', in: *In Workshop on Text Summarization in Conjunction with*

the ACL 2002 and including the DARPA/NIST sponsored DUC 2002 Meeting on Text Summarization, pp. 11–12, Philadelphia, PA, USA, 2002, Online: http://www-nlpir.nist.gov/projects/duc/pubs/2002papers/kuleuven_angheluta.pdf.

Giuseppe Attardi, Stefano Dei Rossi, Giulia Di Pietro, Alessandro Lenci, Simonetta Montemagni, and Maria Simi: 'A Resource and Tool for Super-sense Tagging of Italian Texts', in: *Proceedings of the Seventh International Conference on Language Resources and Evaluation*, LREC 2010, pp. 2242–2248, Valletta, Malta, 2010, Online: http://www.lrec-conf.org/proceedings/lrec2010/pdf/216_Paper.pdf.

Christopher Baker, Arash Shaban-Nejad, Xiao Su, Volker Haarslev, and Greg Butler: 'Semantic Web Infrastructure for Fungal Enzyme Biotechnologists', *Web Semantics: Science, Services and Agents on the World Wide Web* 4 (3): 168–180, 2006, Online: http://www.websemanticsjournal.org/index.php/ps/article/view/94.

Daniel Bär, Chris Biemann, Iryna Gurevych, and Torsten Zesch: 'UKP: Computing Semantic Textual Similarity by Combining Multiple Content Similarity Measures', in: *Proceedings of the 6th International Workshop on Semantic Evaluation, held in conjunction with the 1st Joint Conference on Lexical and Computational Semantics*, pp. 435–440, Montreal, Canada, 2012, Online: http://www.aclweb.org/anthology/S12-1059.

Marco Baroni and Roberto Zamparelli: 'Nouns are vectors, adjectives are matrices: representing adjective-noun constructions in semantic space', in: *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP 2010, pp. 1183–1193, Cambridge, Massachusetts, 2010, Online: https://www.aclweb.org/anthology/D/D10/D10-1115.pdf.

Doug Beeferman, Adam Berger, and John Lafferty: 'Statistical models for text segmentation', *Machine learning* 34 (1): 177–210, 1999, Online: http://link.springer.com/article/10.1023%2FA%3A1007506220214.

Richard Ernest Bellman: *Dynamic programming*, Vol. A Rand Corporation research study, Princeton U.P., 1957, Online: http://press.princeton.edu/titles/9234.html.

Darina Benikova, Chris Biemann, and Marc Reznicek: 'NoSta-D Named Entity Annotation for German: Guidelines and Dataset', in: *Proceedings of the Ninth International Conference on Language Resources and Evaluation*, LREC 2014, Reykjavik, Iceland, 2014, Online: http://www.lrec-conf.org/proceedings/lrec2014/summaries/276.html.

Darina Benikova, Seid Muhie Yimam, and Chris Biemann: 'GermaNER: Free Open German Named Entity Recognition Tool', in: *International Conference of the German Society for Computational Linguistics and Language Technology*, GSCL 2015, pp. 31–38, Essen, Germany, 2015, Online: http://gscl2015.inf.uni-due.de/wp-content/uploads/2015/09/gscl2015-proceedings.pdf.

Lester V. Berry: *Roget's International Thesaurus*, Harper Colophon Books, Crowell, 1962.

Chris Biemann: 'Chinese Whispers: An Efficient Graph Clustering Algorithm and Its Application to Natural Language Processing Problems', in: *Proceedings of the First Workshop on Graph Based Methods for Natural Language Processing held in conjunction with NAACL*, TextGraphs-1, pp. 73–80, New York, NY, USA, 2006, Online: http://www.aclweb.org/anthology/W06-3812.

Chris Biemann: 'Unsupervised Part-of-Speech Tagging in the Large', *Research on Language and Computation* 7 (2-4): 101–135, December 2009, Online: http://link.springer.com/article/10.1007%2Fs11168-010-9067-9.

Chris Biemann: *Structure Discovery in Natural Language*, Theory and Applications of Natural Language Processing, Springer Heidelberg, 2011, Online: http://link.springer.com/book/10.1007%2F978-3-642-25923-4.

Chris Biemann: 'Turk Bootstrap Word Sense Inventory 2.0: A Large-Scale Resource for Lexical Substitution', in: *Proceedings of the Eight International Conference on Language Resources and Evaluation*, LREC 2012, pp. 4038–4042, Istanbul, Turkey, 2012, Online: http://www.lrec-conf.org/proceedings/lrec2012/pdf/252_Paper.pdf.

Chris Biemann and Martin Riedl: 'Text: Now in 2D! A Framework for Lexical Expansion with Contextual Similarity', *Journal of Language Modelling* 1 (1): 55–95, 2013, Online: http://jlm.ipipan.waw.pl/index.php/JLM/article/view/60.

Christopher M. Bishop: *Pattern Recognition and Machine Learning*, Springer-Verlag New York, Inc., 2006, Online: http://www.springer.com/us/book/9780387310732.

Yonatan Bisk and Julia Hockenmaier: 'An HDP Model for Inducing Combinatory Categorial Grammars', *Transactions of the Association for Computational Linguistics* 1: 75–88, 2013, Online: https://tacl2013.cs.columbia.edu/ojs/index.php/tacl/article/view/79.

Olivier Blanc, Matthieu Constant, and Patrick Watrin: 'A Finite-state Super-chunker', in: *Proceedings of the 12th International Conference on Implementation and Application of Automata*, CIAA 2007, pp. 306–308, Prague, Czech Republic, 2007, Online: http://link.springer.com/chapter/10.1007%2F978-3-540-76336-9_29.

David M. Blei and Pedro J. Moreno: 'Topic segmentation with an aspect hidden Markov model', in: *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR 2001, pp. 343–348, New Orleans, Louisiana, USA, 2001, Online: http://doi.acm.org/10.1145/383952.384021.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan: 'Latent dirichlet allocation', *Journal of Machine Learning Research (JMLR)* 3: 993–1022, March 2003, Online: http://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf.

Rens Bod: 'Is the End of Supervised Parsing in Sight?', in: *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, ACL 2007, pp. 400–407, Prague, Czech Republic, 2007, Online: http://www.aclweb.org/anthology/P07-1051.

Olivier Bodenreider: 'The unified medical language system (UMLS): integrating biomedical terminology', *Nucleic acids research* 32 (suppl 1): D267–D270, 2004, Online: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC308795/.

Bernd Bohnet: 'Top Accuracy and Fast Dependency Parsing is Not a Contradiction', in: *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING 2010, pp. 89–97, Beijing, China, 2010, Online: http://www.anthology.aclweb.org/C/C10/C10-1011.pdf.

Stefan Bordag: 'A Comparison of Co-occurrence and Similarity Measures As Simulations of Context', in: *Proceedings of the 9th International Conference on Computational Linguistics and Intelligent Text Processing*, CICLing 2008, pp. 52–63, Haifa, Israel, 2008, Online: http://link.springer.com/content/pdf/10.1007%2F978-3-540-78135-6_5.pdf.

Dhouha Bouamor, Nasredine Semmar, and Pierre Zweigenbaum: 'Identifying bilingual Multi-Word Expressions for Statistical Machine Translation', in: *Proceedings of the Eight International Conference on Language Resources and Evaluation*, LREC 2012, pp. 674–679, Istanbul, Turkey, 2012, Online: http://www.lrec-conf.org/proceedings/lrec2012/pdf/886_Paper.pdf.

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith: 'The TIGER Treebank', in: *Proceedings of the Workshop on Treebanks and Linguistic Theories*, pp. 24–41, Sozopol, Bulgaria, 2002, Online: http://www.bultreebank.org/proceedings/paper03.pdf.

Thorsten Brants and Alex Franz: 'Web 1T 5-gram Corpus Version 1', *Technical report*, Google Research, 2006, Online: https://catalog.ldc.upenn.edu/LDC2006T13.

Ted Briscoe, John Carroll, and Rebecca Watson: 'The Second Release of the RASP System', in: *Proceedings of the COLING/ACL on Interactive Presentation Sessions*, COLING-ACL 2006, pp. 77–80, Sydney, Australia, 2006, Online: http://dx.doi.org/10.3115/1225403.1225423.

Sabine Buchholz and Erwin Marsi: 'CoNLL-X Shared Task on Multilingual Dependency Parsing', in: *Proceedings of the Tenth Conference on Computational Natural Language Learning*, CoNLL-X 2006, pp. 149–164, New York City, New York, 2006, Online: http://www.aclweb.org/anthology/W06-2920.

Chris Buckley and Ellen M. Voorhees: 'Retrieval Evaluation with Incomplete Information', in: *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR 2004, pp. 25–32, Sheffield, United Kingdom, 2004, Online: http://doi.acm.org/10.1145/1008992.1009000.

Ekaterina Buyko and Udo Hahn: 'Evaluating the Impact of Alternative Dependency Graph Encodings on Solving Event Extraction Tasks', in: *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP 2010, pp. 982–992, Cambridge, Massachusetts, 2010, Online: http://www.aclweb.org/anthology/D10-1096.

Guillaume Cabanac, Gilles Hubert, Mohand Boughanem, and Claude Chrisment: 'Tie-breaking Bias: Effect of an Uncontrolled Parameter on Information Retrieval Evaluation', in: *Conference on Multilingual and Multimodal Information Access Evaluation* , CLEF, pp. 112–123, Padua, Italy, 2010, Online: http://www.irit.fr/publis/SIG/2010_CLEF_CHBC.pdf.

Sander Canisius, Toine Bogers, Antal van den Bosch, Jeroen Geertzen, and Erik Tjong Kim Sang: 'Dependency Parsing by Inference over High-recall Dependency Predictions', in: *Proceedings of the Tenth Conference on Computational Natural Language Learning*, CoNLL-X 2006, pp. 176–180, New York City, New York, 2006, Online: https://aclweb.org/anthology/W/W06/W06-2924.pdf.

Richard Eckart de Castilho and Iryna Gurevych: 'A broad-coverage collection of portable NLP components for building shareable analysis pipelines', in: *Proceedings of the Workshop on Open Infrastructures and Analysis Frameworks for HLT at COLING 2014*, OIAF4HLT, pp. 1–11, Dublin, Ireland, 2014, Online: http://glicom.upf.edu/OIAF4HLT/pdf/OIAF4HLT01.pdf.

Jonathan Chang, Jordan Boyd-Graber, Chong Wang, Sean Gerrish, and David M. Blei: 'Reading Tea Leaves: How Humans Interpret Topic Models', in: *Proceedings of the Conference on Neural Information Processing Systems*, NIPS 2009, pp. 288–296, Vancouver, British Columbia, 2009, Online: http://papers.nips.cc/paper/3700-reading-tea-leaves-how-humans-interpret-topic-models.

Wenliang Chen, Yue Zhang, and Min Zhang: 'Feature Embedding for Dependency Parsing', in: *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, COLING 2014, pp. 816–826, Dublin, Ireland, 2014, Online: http://www.aclweb.org/anthology/C14-1078.

Freddy Y. Y. Choi: 'Advances in domain independent linear text segmentation', in: *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, NAACL 2001, pp. 26–33, Seattle, WA, USA, 2000, Online: http://www.aclweb.org/anthology/A00-2004.

Freddy Yu Yan Choi, Peter Wiemer-Hastings, and Johanna Moore: 'Latent Semantic Analysis for Text Segmentation', in: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP 2001, pp. 109–117, Pittsburgh, PA, USA, 2001, Online: http://www.cs.cornell.edu/home/llee/emnlp/papers/choi.pdf.

Miranda Chong and Lucia Specia: 'Lexical Generalisation for Word-level Matching in Plagiarism Detection.', in: *Proceedings of the Conference on Recent Advances in Natural Language Processing*, RANLP 2011, pp. 704–709, Hissar, Bulgaria, 2011, Online: http://www.aclweb.org/anthology/R11-1102.

Stefan Thomas Christian Hänig, Stefan Bordag: 'Modular Classifier Ensemble Architecture for Named Entity Recognition on Low Resource Systems', in: *Proceedings of the KONVENS GermEval Shared Task on Named Entity Recognition*, pp. 113–116, Hildesheim, Germany, 2014.

Kenneth Ward Church and Patrick Hanks: 'Word association norms, mutual information, and lexicography', *Computational Linguistics* 16 (1): 22–29, 1990, Online: http://www.aclweb.org/anthology/J90-1003.

Massimiliano Ciaramita and Mark Johnson: 'Supersense Tagging of Unknown Nouns in WordNet', in: *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, EMNLP 2003, pp. 168–175, Sapporo, Japan, 2003, Online: http://www.anthology.aclweb.org/W/W03/W03-1022.pdf.

Alexander Clark: 'Inducing Syntactic Categories by Context Distribution Clustering', in: *Fourth Conference on Computational Natural Language Learning and of the Second Learning Language in Logic Workshop*, pp. 91–94, Lisboa, Portugal, 2000, Online: http://www.aclweb.org/anthology/W00-0717.

Michael Collins: 'Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms', in: *Proceedings of the Conference on Empirical methods in Natural Language Processing*, EMNLP 2002, pp. 1–8, Philadelphia, PA, USA, 2002, Online: www.aclweb.org/anthology/W02-1001.pdf.

Bart Cramer: 'Limitations of Current Grammar Induction Algorithms', in: *Proceedings of the Annual Meeting of the Association of Computational Linguistics - Student Research Workshop*, ACL 2007, pp. 43–48, Prague, Czech Republic, 2007, Online: http://www.aclweb.org/anthology/P07-3008.

David Crystal: *A Dictionary of Linguistics and Phonetics*, Blackwell, 6 edition, 2008, Online: http://eu.wiley.com/WileyCDA/WileyTitle/productCd-1405152966.html.

James R. Curran: 'Ensemble methods for automatic thesaurus extraction', in: *Proceedings of the ACL-02 conference on Empirical methods in natural language processing*, EMNLP 2002, pp. 222–229, Philadelphia, PA, USA, 2002, Online: http://www.aclweb.org/anthology/W02-1029.

James R. Curran: *From Distributional to Semantic Similarity*, Ph.D. thesis, University of Edinburgh, 2004, Online: http://sydney.edu.au/engineering/it/~james/pubs/pdf/phdthesis.pdf.

James R. Curran and Marc Moens: 'Improvements in Automatic Thesaurus Extraction', in: *Proceedings of the ACL-02 Workshop on Unsupervised Lexical Acquisition - Volume 9*, ULA 2002, pp. 59–66, Philadelphia, PA, USA, 2002, Online: www.aclweb.org/anthology/W02-0908.

Ido Dagan, Dan Roth, Mark Sammons, and Fabio M. Zanzotto: 'Recognizing Textual Entailment: Models and Applications', *Synthesis Lectures on Human Language Technologies* 6 (4): 1–220, 2013, Online: http://www.morganclaypool.com/doi/abs/10.2200/S00509ED1V01Y201305HLT023.

Béatrice Daille, Éric Gaussier, and Jean-Marc Langé: 'Towards Automatic Extraction of Monolingual and Bilingual Terminology', in: *Proceedings of the 15th Conference on Computational Linguistics - Volume 1*, COLING 1994, pp. 515–521, Kyoto, Japan, 1994, Online: http://dx.doi.org/10.3115/991886.991975.

Jeffrey Dean and Sanjay Ghemawat: 'MapReduce: Simplified Data Processing on Large Clusters', in: *Proceedings of Operating Systems, Design & Implementation '04*, OSDI, pp. 137–150, San Francisco, CA, USA, 2004, Online: http://research.google.com/archive/mapreduce.html.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman: 'Indexing by Latent Semantic Analysis', *Journal of the American Society for Information Science* 41 (6): 391–407, 1990, Online: http://lsa.colorado.edu/papers/JASIS.lsi.90.pdf.

Mark Dredze, John Blitzer, Partha Pratim Talukdar, Kuzman Ganchev, João Graca, and Fernando Pereira: 'Frustratingly Hard Domain Adaptation for Dependency Parsing', in: *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pp. 1051–1055, Prague, Czech Republic, June 2007, Online: http://www.aclweb.org/anthology/D/D07/D07-1112.pdf.

Lan Du, Wray Buntine, and Huidong Jin: 'A segmented topic model based on the two-parameter Poisson-Dirichlet process', *Machine Learning* 81 (1): 5–19, 2010, Online: http://link.springer.com/article/10.1007%2Fs10994-010-5197-4.

Lan Du, Wray Buntine, and Mark Johnson: 'Topic Segmentation with a Structured Topic Model', in: *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-HlT 2013, pp. 190–200, Atlanta, GA, USA, 2013, Online: http://www.aclweb.org/anthology/N13-1019.

David Dubin: 'The Most Influential Paper Gerard Salton Never Wrote.', *Library Trends* 52 (4): 748–764, 2004, Online: https://www.ideals.illinois.edu/bitstream/handle/2142/1697/Dubin748764.pdf?sequence=2.

Ted Dunning: 'Accurate methods for the statistics of surprise and coincidence', *Computational Linguistics* 19 (1): 61–74, March 1993, Online: http://www.aclweb.org/anthology/J93-1003.

Jacob Eisenstein: 'Hierarchical text segmentation from multi-scale lexical cohesion', in: *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL-HLT 2009, pp. 353–361, Boulder, CO, USA, 2009, Online: http://www.aclweb.org/anthology/N09-1040.

Stefan Evert: *The Statistics of Word Cooccurrences: Word Pairs and Collocations*, Ph.D. thesis, Institut für maschinelle Sprachverarbeitung, University of Stuttgart, 2005, Online: http://elib.uni-stuttgart.de/opus/volltexte/2005/2371/pdf/Evert2005phd.pdf.

Stefan Evert: 'A lexicographic evaluation of German adjective-noun collocations', in: *Proceedings of the LREC 2008 Workshop Towards a Shared Task for Multiword Expressions*, MWE 2008, pp. 3–6, Marrakech, Morocco, 2008, Online: http://www.stefan-evert.de/PUB/Evert2008_MWE_Resource.pdf.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin: 'LIBLINEAR: A Library for Large Linear Classification', *Journal of Machine Learning Research* 9: 1871–1874, 2008, Online: http://www.jmlr.org/papers/volume9/fan08a/fan08a.pdf.

Katja Filippova and Michael Strube: 'Dependency Tree Based Sentence Compression', in: *5th International Natural Language Generation*, INLG 2008, 2008, Online: http://www.aclweb.org/anthology/W08-1105.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin: 'Placing search in context: the concept revisited', in: *Proceedings of the 10th international conference on World Wide Web*, WWW 2001, pp. 406–414, Hong Kong, Hong Kong, 2001, Online: http://doi.acm.org/10.1145/371920.372094.

John Rupert Firth: 'A synopsis of linguistic theory 1930-1955', *Studies in linguistic analysis* pp. 1–32, 1957, Online: http://annabellelukin.edublogs.org/files/2013/08/Firth-JR-1962-A-Synopsis-of-Linguistic-Theory-wfihi5.pdf.

Pavlina Fragkou, Vassilios Petridis, and Athanasios Kehagias: 'A Dynamic Programming Algorithm for Linear Text Segmentation', *Journal of Intelligent Information Systems* 23 (2): 179–197, 2004, Online: http://link.springer.com/article/10.1023/B%3AJIIS.0000039534.65423.00.

Winthrop Nelson Francis and Henry Kučera: *A Standard Corpus of Present-Day Edited American English, for use with Digital Computers*, Brown University, 1964, Online: http://clu.uni.no/icame/brown/bcm.html.

Katerina T. Frantzi, Sophia Ananiadou, and Jun-ichi Tsujii: 'The C-value/NC-value Method of Automatic Recognition for Multi-Word Terms', in: *Proceedings of the Second European Conference on Research and Advanced Technology for Digital Libraries*, ECDL 1998, pp. 585–604, Heraklion, Greece, 1998, Online: http://link.springer.com/chapter/10.1007%2F3-540-49653-X_35.

Michel Galley, Kathleen McKeown, Eric Fosler-Lussier, and Hongyan Jing: 'Discourse Segmentation of Multi-Party Conversation', in: *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, ACL 2003, pp. 562–569, Sapporo, Japan, 2003, Online: http://aclweb.org/anthology/P/P03/P03-1071.pdf.

Rashmi Gangadharaiah, Ralf D. Brown, and Jaime Carbonell: 'Monolingual Distributional Profiles for Word Substitution in Machine Translation', in: *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, COLING 2010, pp. 320–328, Beijing, China, 2010, Online: http://www.aclweb.org/anthology/C10-2037.pdf.

Maria Georgescul, Alexander Clark, and Susan Armstrong: 'An Analysis of Quantitative Aspects in the Evaluation of Thematic Segmentation Algorithms', in: *Proceedings of the 7th SIGdial Workshop on Discourse and Dialogue*, pp. 144–151, Sydney, Australia, July 2006, Online: http://www.aclweb.org/anthology/W06-1320.

Hristo Georgiev: *English Algorithmic Grammar*, Bloomsbury Academic, 2006, Online: http://www.bloomsbury.com/uk/english-algorithmic-grammar-9781847143358/.

Eugenie Giesbrecht and Stefan Evert: 'Part-of-speech tagging - a solved task? An evaluation of POS taggers for the Web as corpus', in: *Proceedings of the 5th Web as Corpus Workshop*, WAC5, pp. 29–35, San Sebastian, Spain, 2009, Online: https://www.sigwac.org.uk/attachment/wiki/WAC5/WAC5_proceedings.pdf.

Jennifer Gillenwater, Kuzman Ganchev, João Graça, Fernando Pereira, and Ben Taskar: 'Sparsity in dependency grammar induction', in: *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, ACL 2010, pp. 194–199, Uppsala, Sweden, 2010, Online: https://aclweb.org/anthology/P/P10/P10-2036.pdf.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith: 'Part-of-speech tagging for Twitter: annotation, features, and experiments', in: *Proceedings of the*

49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, ACL-HLT 2011, pp. 42–47, Portland, OR, USA, 2011, Online: http://aclweb.org/anthology/P/P11/P11-2008.pdf.

Claudio Giuliano, Alfio Gliozzo, and Carlo Strapparava: 'FBK-irst: Lexical Substitution Task Exploiting Domain and Syntagmatic Coherence', in: *Proceedings of the 4th International Workshop on Semantic Evaluations*, SemEval 2007, pp. 145–148, Prague, Czech Republic, 2007, Online: http://www.aclweb.org/anthology/S07-1029.

Alfio Gliozzo, Chris Biemann, Martin Riedl, Bonaventura Coppola, Michael R. Glass, and Matthew Hatem: 'JoBimText Visualizer: A Graph-based Approach to Contextualizing Distributional Similarity', in: *Proceedings of TextGraphs-8 Graph-based Methods for Natural Language Processing held in conjunction with EMNLP '13*, TextGraphs-8, pp. 6–10, Seattle, WA, USA, 2013, Online: http://aclweb.org/anthology/W13-5002.

Yoav Goldberg and Jon Orwant: 'A Dataset of Syntactic-Ngrams over Time from a Very Large Corpus of English Books', in: *Second Joint Conference on Lexical and Computational Semantics , Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, *SEM, pp. 241–247, Atlanta, GA, USA, 2013, Online: http://www.aclweb.org/anthology/S13-1035.

Gene H. Golub and William M. Kahan: 'Calculating the singular values and pseudo-inverse of a matrix', *Journal of the Society for Industrial and Applied Mathematics - Series B: Numerical Analysis* 2: 205–224, 1965, Online: http://epubs.siam.org/doi/abs/10.1137/0702016.

James Gorman and James R. Curran: 'Scaling distributional similarity to large corpora', in: *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL 2006, pp. 361–368, Sydney, Australia, 2006, Online: http://www.aclweb.org/anthology/P06-1046.

Amit Goyal and Hal Daumé, III: 'Generating semantic orientation lexicon using large data and thesaurus', in: *Proceedings of the 2nd Workshop on Computational Approaches to Subjectivity and Sentiment Analysis*, WASSA 2011, pp. 37–43, Portland, OR, USA, 2011, Online: http://www.aclweb.org/anthology/W11-1705.

Amit Goyal, Jagadeesh Jagarlamudi, Hal Daumé, III, and Suresh Venkatasubramanian: 'Sketch techniques for scaling distributional similarity to the web', in: *Proceedings of the 2010 Workshop on GEometrical Models of Natural Language Semantics*, GEMS 2010, pp. 51–56, Uppsala, Sweden, 2010, Online: http://dl.acm.org/citation.cfm?id=1870516.1870524.

Thomas L. Griffiths and Mark Steyvers: 'Finding Scientific Topics', *Proceedings of the National Academy of Sciences* 101: 5228–5235, 2004, Online: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC387300/.

Amit Gruber, Michal Rosen-zvi, and Yair Weiss: 'Hidden topic Markov models', in: *Proceedings of Artificial Intelligence and Statistics*, pp. 163–170, San Juan, Puerto Rico, 2007, Online: http://jmlr.csail.mit.edu/proceedings/papers/v2/gruber07a/gruber07a.pdf.

Weiwei Guo and Mona Diab: 'Semantic Topic Models: Combining Word Distributional Statistics and Dictionary Definitions', in: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP 2011, pp. 552–561, Edinburgh, United Kingdom, 2011, Online: https://aclweb.org/anthology/D/D11/D11-1051.pdf.

Michael Alexander Kirkwood Halliday: 'Towards a language-based theory of learning', *Linguistics and education* 5 (2): 93–116, 1993, Online: http://lchc.ucsd.edu/mca/Paper/JuneJuly05/HallidayLangBased.pdf.

Birgit Hamp and Helmut Feldweg: 'GermaNet - a Lexical-Semantic Net for German', in: *Proceedings of ACL workshop Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications*, pp. 9–15, Madrid, Spain, 1997, Online: https://www.aclweb.org/anthology/W/W97/W97-0802.pdf.

Bo Han and Timothy Baldwin: 'Lexical Normalisation of Short Text Messages: Makn Sens a #Twitter', in: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, ACL-HLT 2011, pp. 368–378, Portland, OR, USA, 2011, Online: http://www.aclweb.org/anthology/P11-1038.

Zellig Sabbetai Harris: *Methods in Structural Linguistics*, University of Chicago Press, Chicago, 1951.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman: *The Elements of Statistical Learning*, Springer, 2009, Online: http://statweb.stanford.edu/~tibs/ElemStatLearn/.

William P. Headden III, Mark Johnson, and David McClosky: 'Improving unsupervised dependency parsing with richer contexts and smoothing', in: *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL-HLT 2009, pp. 101–109, Boulder, CO, USA, 2009, Online: http://www.aclweb.org/anthology/N09-1012.

Marti A. Hearst: 'Multi-paragraph segmentation of expository text', in: *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, pp. 9–16, Las Cruces, NM, USA, 1994, Online: https://aclweb.org/anthology/P/P94/P94-1002.pdf.

Marti A. Hearst: 'TextTiling : Segmenting Text into Multi-paragraph Subtopic Passages', *Computational Linguistics* 23 (1): 33–64, 1997, Online: https://aclweb.org/anthology/J/J97/J97-1003.pdf.

Gregor Heinrich: 'Parameter estimation for text analysis', *Technical report*, Technical Note vsonix GmbH and University of Leipzig, Germany, 2004, Online: www.arbylon.net/publications/text-est.pdf.

Gregor Heinrich: 'Typology of Mixed-Membership Models: Towards a Design Method', in: *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, ECML/PKDD 2011 Vol. 6912, pp. 32–47, Athens, Greece, 2011, Online: http://link.springer.com/content/pdf/10.1007%2F978-3-642-23783-6_3.pdf.

Karl Moritz Hermann and Phil Blunsom: 'Multilingual Models for Compositional Distributed Semantics', in: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, ACL 2014, pp. 58–68, Baltimore, MA, USA, 2014, Online: http://aclweb.org/anthology/P14-1006.

Felix Hill, Roi Reichart, and Anna Korhonen: 'SimLex-999: Evaluating Semantic Models with (Genuine) Similarity Estimation', *Computational Linguistics* 2014, Online: http://www.cl.cam.ac.uk/~fh295/simlex.html.

Donald Hindle: 'Noun Classification from Predicate-argument Structures', in: *Proceedings of the 28th Annual Meeting on Association for Computational Linguistics*, ACL 1990, pp. 268–275, Pittsburgh, PA, USA, 1990, Online: https://aclweb.org/anthology/P/P90/P90-1034.pdf.

Lynette Hirschman and Rob Gaizauskas: 'Natural Language Question Answering: The View from Here', *Journal of Natural Language Engineering (NLE)* 7 (4): 275–300, 2001, Online: http://dx.doi.org/10.1017/S1351324901002807.

Thomas Hofmann: 'Probabilistic latent semantic indexing', in: *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR 1999, pp. 50–57, New York, NY, USA, 1999, Online: http://arxiv.org/pdf/1301.6705.

Thomas Hofmann: 'Unsupervised Learning by Probabilistic Latent Semantic Analysis', *Machine Learning* 42 (1-2): 177–196, 2001, Online: http://link.springer.com/article/10.1023%2FA%3A1007617005950.

Michael Holmes, Alexander Gray, and Charles Isbell: 'Fast SVD for Large-Scale Matrices', in: *Workshop on Efficient Machine Learning at NIPS*, pp. 1–2, Vancouver, British Columbia, Canada, 2007, Online: sysrun.haifa.il.ibm.com/hrl/bigml/files/Holmes.pdf.

Paul Jaccard: 'The Distribution of the Flora in the Alpine Zone', *New Phytologist* 11 (2): 37–50, February 1912, Online: http://onlinelibrary.wiley.com/doi/10.1111/j.1469-8137.1912.tb05611.x/pdf.

Mario Jarmasz and Stan Szpakowicz: 'The Design and Implementation of an Electronic Lexical Knowledge Base', in: *Proceedings of the 14th Biennial Conference of the Canadian Society on Computational Studies of Intelligence: Advances in Artificial Intelligence*, AI 2001, pp. 325–334, Ottawa, Canada, 2001, Online: http://dl.acm.org/citation.cfm?id=647462.726295.

Richard Johansson and Pierre Nugues: 'The Effect of Syntactic Representation on Semantic Role Labeling', in: *Proceedings of the 22Nd International Conference on Computational Linguistics*, COLING 2008, pp. 393–400, Manchester, United Kingdom, 2008, Online: http://www.aclweb.org/anthology/C/C08/C08-1050.pdf.

Daniel Jurafsky and James H Martin: *Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics*, Pearson International Edition, 2009, Online: https://www.cs.colorado.edu/~martin/slp.html.

John S. Justeson and Slava M. Katz: 'Technical terminology: some linguistic properties and an algorithm for identification in text', *Natural Language Engineering* 1: 9–27, 3 1995, Online: http://journals.cambridge.org/article_S1351324900000048.

Friedrich Wilhelm Kaeding: *Häufigkeitswörterbuch der deutschen Sprache*, E.S. Mittler & Sohn, 1898, Online: https://books.google.de/books?id=PHEVAAAAYAAJ.

Graham Katz and Eugenie Giesbrecht: 'Automatic Identification of Non-compositional Multi-word Expressions Using Latent Semantic Analysis', in: *Proceedings of the Workshop on Multiword Expressions: Identifying and Exploiting Underlying Properties*, MWE 2006, pp. 12–19, Sydney, Australia, 2006, Online: anthology.aclweb.org/W/W06/W06-1203.pdf.

Sukhpreet Kaur and Kamaljeet Kaur Mangat: 'Comparative Analysis of C99 and TopicTiling Text Segmentation Algorithms', *International Journal of Research in Engineering and Technology (IJRET)* 2 (9): 529–534, 2013, Online: http://doi.org/10.15623/ijret.2013.0209080.

Anna Kazantseva and Stan Szpakowicz: 'Topical Segmentation: A Study of Human Performance and a New Measure of Quality', in: *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT 2012, pp. 211–220, Montreal, Canada, 2012, Online: http://www.aclweb.org/anthology/N12-1022.

Douwe Kiela and Stephen Clark: 'A Systematic Study of Semantic Vector Space Model Parameters', in: *Proceedings of the 2nd Workshop on Continuous Vector Space Models and their Compositionality in conjunction with the EACL '14*, CVSC, pp. 21–30, Gothenburg, Sweden, 2014, Online: http://aclweb.org/anthology/W14-1503.

Adam Kilgarriff, Pavel Rychlý, and David Tugwell Pavel Smrz: 'The Sketch Engine', in: *Proceedings of the 11th EURALEX International Congress*, pp. 105–115, Lorient, France,

2004, Online: http://www.euralex.org/elx_proceedings/Euralex2004/011_2004_V1_Adam%20KILGARRIFF,%20Pavel%20RYCHLY,%20Pavel%20SMRZ,%20David%20TUGWELL_The%20%20Sketch%20Engine.pdf.

Jin-Dong Kim, Tomoko Ohta, Yuka Tateisi, and Jun ichi Tsujii: 'GENIA corpus - a semantically annotated corpus for bio-textmining', *Bioinformatics* 19 (suppl 1): i180–182, 2003, Online: http://bioinformatics.oxfordjournals.org/content/19/suppl_1/i180.abstract.

Kazuaki Kishida: 'Property of average precision and its generalization: An examination of evaluation indicator for information retrieval experiment', *Technical report*, Surugadai University, 2005, Online: www.nii.ac.jp/TechReports/05-014E.pdf.

Dan Klein and Christopher D. Manning: 'A generative constituent-context model for improved grammar induction', in: *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL 2002, pp. 128–135, Philadelphia, PA,USA, 2002, Online: https://aclweb.org/anthology/P/P02/P02-1017.pdf.

Dan Klein and Christopher D. Manning: 'Corpus-based induction of syntactic structure: Models of dependency and constituency', in: *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, ACL 2004, pp. 478–485, Barcelona, Spain, 2004, Online: https://aclweb.org/anthology/P/P04/P04-1061.pdf.

Ioannis Korkontzelos: *Unsupervised Learning of Multiword Expressions*, Ph.D. thesis, University of York, UK, 2010, Online: http://etheses.whiterose.ac.uk/2091/1/IoannisKorkontzelos-PhDThesis.pdf.

Gerhard Kremer, Katrin Erk, Sebastian Padó, and Stefan Thater: 'What Substitutes Tell Us - Analysis of an "All-Words" Lexical Substitution Corpus', in: *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, EACL 2014, pp. 540–549, Gothenburg, Sweden, 2014, Online: https://aclweb.org/anthology/E/E14/E14-1057.pdf.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira: 'Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data', in: *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML 2001, pp. 282–289, Williams College, Williamstown, MA, USA, 2001, Online: http://dl.acm.org/citation.cfm?id=645530.655813.

Adam Lally, Sugato Bagchi, Michael Barborak, David Buchanan, Jennifer Chu-Carroll, David Ferrucci, Michael Glass, Aditya Kalyanpur, Erik Mueller, James William Murdock, Siddharth Patwardhan, John M. Prager, and Christopher A. Welty: 'WatsonPaths: Scenario-based Question Answering and Inference over Unstructured Information', *IBM Research*

Report RC25489 (WAT1409-048), IBM T. J. Watson Research Center, 2014, Online: http://www.patwardhans.net/papers/LallyEtAl14.pdf.

Sylvain Lamprier, Tassadit Amghar, Bernard Levrat, and Frederic Saubion: 'On Evaluation Methodologies for Text Segmentation Algorithms', in: *Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence*, ICTAI 2007, pp. 19–26, Patras, Greece, 2007, Online: http://dx.doi.org/10.1109/ICTAI.2007.142.

J. Richard Landis and Gary G. Koch: 'The measurement of observer agreement for categorical data', *Biometrics* 33 (1): 159–174, 1977, Online: http://www.ncbi.nlm.nih.gov/pubmed/843571.

Jey Han Lau, Paul Cook, Diana McCarthy, David Newman, and Timothy Baldwin: 'Word Sense Induction for Novel Sense Detection', in: *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, EACL 2012, pp. 591–601, Avignon, France, 2012, Online: http://www.aclweb.org/anthology/E12-1060.

Lillian Lee: 'Measures of distributional similarity', in: *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, ACL 1999, pp. 25–32, College Park, MA, USA, 1999, Online: http://dx.doi.org/10.3115/1034678.1034693.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Chris Bizer: 'DBpedia - A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia', *Semantic Web Journal* 6 (2): 167–195, 2015, Online: http://www.semantic-web-journal.net/content/dbpedia-large-scale-multilingual-knowledge-base-extracted-wikipedia-0.

Omer Levy and Yoav Goldberg: 'Dependency-Based Word Embeddings', in: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 302–308, Baltimore, Maryland, 2014b, Online: http://www.aclweb.org/anthology/P14-2050.

Omer Levy and Yoav Goldberg: 'Neural Word Embedding as Implicit Matrix Factorization', in: *Advances in Neural Information Processing Systems 27*, NIPS 2014, pp. 2177–2185, 2014a, Online: http://papers.nips.cc/paper/5477-neural-word-embedding-as-implicit-matrix-factorization.pdf.

Baoli Li and Liping Han: 'Distance Weighted Cosine Similarity Measure for Text Classification', in: *Intelligent Data Engineering and Automated Learning – IDEAL 2013*, Lecture Notes in Computer Science Vol. 8206, pp. 611–618, 2013, Online: http://dx.doi.org/10.1007/978-3-642-41278-3_74.

Dekang Lin: 'Using syntactic dependency as local context to resolve word sense ambiguity', in: *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, ACL 1998/EACL 1997, pp. 64–71, Madrid, Spain, 1997, Online: http://www.aclweb.org/anthology/P97-1009.

Dekang Lin: 'Automatic retrieval and clustering of similar words', in: *Proceedings of the 17th international conference on Computational linguistics*, COLING 1998, pp. 768–774, Montreal, Quebec, Canada, 1998, Online: http://dx.doi.org/10.3115/980432.980696.

Jimmy Lin and Chris Dyer: *Data-Intensive Text Processing with MapReduce*, Morgan & Claypool Publishers, San Rafael, CA, 2010, Online: http://www.morganclaypool.com/doi/pdf/10.2200/S00274ED1V01Y201006HLT007.

Fernando Llopis, Antonio Ferrández Rodríguez, and José Luis Vicedo González: 'Text Segmentation for Efficient Information Retrieval.', in: *Proceedings of the Third International Conference on Intelligent Text Processing and Computational Linguistics*, CICLing 2002, pp. 373–380, Mexico City, Mexico, 2002, Online: http://link.springer.com/content/pdf/10.1007/3-540-45715-1_39.pdf.

Juan Antonio Lossio-Ventura, Clement Jonquet, Mathieu Roche, and Maguelonne Teisseire: 'Yet Another Ranking Function for Automatic Multiword Term Extraction', in: *Proceedings of the 9th International Conference on Natural Language Processing*, PolTAL 2014, pp. 52–64, Warsaw, Poland, 2014, Online: http://www.lirmm.fr/~jonquet/publications/documents/Article_PolTAL2014_Lossio.pdf.

Bertil Malmberg: *New Trends in Linguistics: An Orientation*, Holmiae Lundinum Gothorum, Stockholm, 1964, Online: https://books.google.de/books?id=O60kAQAAMAAJ.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze: *Introduction to Information Retrieval*, Cambridge University Press, 2008, Online: http://nlp.stanford.edu/IR-book/pdf/irbookonlinereading.pdf.

Christopher D. Manning and Hinrich Schütze: *Foundations of statistical natural language processing*, MIT Press, 1999, Online: http://nlp.stanford.edu/fsnlp/.

Mitchell P. Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert Macintyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger: 'The Penn Treebank: Annotating predicate argument structure', in: *Proceedings of the workshop on Human Language Technology*, pp. 114–119, Plainsboro, NJ, USA, 1994, Online: ftp://ftp.cis.upenn.edu/pub/treebank/doc/arpa94.ps.gz.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini: 'Building a large annotated corpus of English: The Penn Treebank', *Computational linguistics* 19 (2): 313–330, 1993, Online: ftp://ftp.cis.upenn.edu/pub/treebank/doc/arpa94.ps.gz.

David Marecek and Milan Straka: 'Stop-probability estimates computed on a large corpus improve Unsupervised Dependency Parsing', in: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pp. 281–290, Sofia, Bulgaria, 2013, Online: https://ufal.mff.cuni.cz/~marecek/papers/2013_acl_udp.pdf.

Marie-Catherine de Marneffe, Bill Maccartney, and Christopher D. Manning: 'Generating typed dependency parses from phrase structure parses', in: *Proceedings of the International Conference on Language Resources and Evaluation*, LREC 2006, pp. 449–454, Genova, Italy, 2006, Online: http://www.lrec-conf.org/proceedings/lrec2006/pdf/440_pdf.pdf.

Marie-Catherine de Marneffe and Christopher D. Manning: *Stanford typed dependencies manual*, Stanford University, 2015, Online: http://nlp.stanford.edu/software/dependencies_manual.pdf.

Diana McCarthy and Roberto Navigli: 'The English lexical substitution task.', *Language Resources and Evaluation* 43 (2): 139–159, 2009, Online: http://wwwusers.di.uniroma1.it/~navigli/pubs/LRE_2009_McCarthy_Navigli.pdf.

Michael C. McCord, J. William Murdock, and Branimir K. Boguraev: 'Deep Parsing in Watson', *IBM Journal of Research and Development* 56 (3): 264–278, 2012, Online: http://brenocon.com/watson_special_issue/03%20Deep%20parsing.pdf?cm_mc_uid=74185062247214278793382&cm_mc_sid_50200000=1453286269.

Oren Melamud, Jonathan Berant, Ido Dagan, Jacob Goldberger, and Idan Szpektor: 'A Two Level Model for Context Sensitive Inference Rules.', in: *Proceedings of the 51th Annual Meeting of the Association for Computational Linguistics*, ACL 2013, pp. 1331–1340, Sofia, Bulgaria, 2013, Online: http://www.aclweb.org/anthology/P13-1131.

Merriam-Webster: *Merriam-Webster's dictionary of English usage*, Merriam-Webster, Springfield (Mass.), 1994, Online: http://opac.inria.fr/record=b1122273.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean: 'Efficient Estimation of Word Representations in Vector Space', in: *Proceedings of the International Conference on Machine Learning*, ICLR 2013, pp. 1310–1318, Scottsdale, Arizona, USA, 2013, Online: http://arxiv.org/pdf/1301.3781.

George A. Miller: 'WordNet: A Lexical Database for English', *Communications of the ACM* 38: 39–41, 1995, Online: https://mitpress.mit.edu/index.php?q=books/wordnet.

George A. Miller and Walter G. Charles: 'Contextual correlates of semantic similarity', *Language and Cognitive Processes* 6 (1): 1–28, 1991, Online: http://dx.doi.org/10.1080/01690969108406936.

Tristan Miller, Chris Biemann, Torsten Zesch, and Iryna Gurevych: 'Using Distributional Similarity for Lexical Expansion in Knowledge based Word Sense Disambiguation', in: *Proceedings of the 24th International Conference on Computational Linguistics*, COLING 2012, pp. 1781–1796, Mumbai, India, 2012, Online: http://aclweb.org/anthology/C/C12/C12-1109.pdf.

Hemant Misra, François Yvon, Joemon M. Jose, and Olivier Cappe: 'Text Segmentation via Topic Modeling: An Analytical Study', in: *Proceeding of the 18th ACM Conference on Information and Knowledge Management*, pp. 1553–1556, Hong Kong, 2009, Online: http://dl.acm.org/citation.cfm?id=1646170.

Sunny Mitra, Ritwik Mitra, Suman Kalyan Maity, Martin Riedl, Chris Biemann, Pawan Goyal, and Animesh Mukherjee: 'An automatic approach to identify word sense changes in text media across timescales', *Natural Language Engineering* 21: 773–798, 8 2015, Online: http://journals.cambridge.org/article_S135132491500011X.

Sunny Mitra, Ritwik Mitra, Martin Riedl, Chris Biemann, Animesh Mukherjee, and Pawan Goyal: 'That's sick dude!: Automatic identification of word sense change across different timescales', in: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, ACL 2014, pp. 1020–1029, Baltimore, MA, USA, 2014, Online: http://www.aclweb.org/anthology/P14-1096.

Yusuke Miyao, Rune Sætre, Kenji Sagae, Takuya Matsuzaki, and Jun'ichi Tsujii: 'Task-oriented Evaluation of Syntactic Parsers and Their Representations', in: *Proceeding of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, ACL-08:HLT, pp. 46–54, Columbus, OH, USA, 2008, Online: https://aclweb.org/anthology/P/P08/P08-1006.pdf.

Yasaman Motazedi, Mark Dras, and François Lareau: 'Is Bad Structure Better Than No Structure?: Unsupervised Parsing for Realisation Ranking', in: *Proceedings of the 24th International Conference on Computational Linguistics*, COLING 2012, pp. 1811–1830, Mumbai,India, 2012, Online: https://aclweb.org/anthology/C/C12/C12-1111.pdf.

Paul van Mulbregt, Ira Carp, Lawrence Gillick, Steve Lowe, and Jon Yamron: 'Text segmentation and topic tracking on broadcast news via a hidden Markov model approach', in: *Proceedings of 5th International Conference on Spoken Language Processing*, Sydney, Australia, 1998, Online: http://www.isca-speech.org/archive/icslp_1998/i98_0116.html.

Kevin P. Murphy: *Machine Learning: A Probabilistic Perspective*, The MIT Press, 2012, Online: https://mitpress.mit.edu/books/machine-learning-0.

David Nadeau and Satoshi Sekine: 'A survey of named entity recognition and classification', *Linguisticae Investigationes* 30 (1): 3–26, 2007, Online: http://www.ingentaconnect.com/content/jbp/li/2007/00000030/00000001/art00002.

Hiroshi Nakagawa and Tatsunori Mori: 'A Simple but Powerful Automatic Term Extraction Method', in: *International Workshop on Computational Terminology held in conjunction with COLING-02*, COMPUTERM 2002, pp. 1–7, Taipei, Taiwan, 2002, Online: http://dx.doi.org/10.3115/1118771.1118778.

Hiroshi Nakagawa and Tatsunori Mori: 'Automatic term recognition based on statistics of compound nouns and their components', *Terminology* 9 (2): 201–219, 2003, Online: http://dx.doi.org/10.1075/term.9.2.04nak.

Tahira Naseem, Harr Chen, Regina Barzilay, and Mark Johnson: 'Using universal linguistic knowledge to guide grammar induction', in: *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP 2010, pp. 1234–1244, Cambridge, MA, USA, 2010, Online: https://aclweb.org/anthology/D/D10/D10-1120.pdf.

Joakim Nivre and Sandra Kübler: 'Dependency Parsing', in: *Tutorial at COLING-ACL*, Sydney, Australia, 2006, Online: http://stp.lingfil.uu.se/~nivre/docs/ACLslides.pdf?

Adolf Noreen: *Vårt språk: nysvensk grammatik i utförlig framställning*, Lund, C. W. K. Gleerup, Sweden, 1906, Online: https://archive.org/stream/vrtsprknysvenskg02nore/vrtsprknysvenskg02nore_djvu.txt.

Charles Kay Ogden and Ivor Armstrong Richards: *The meaning of meaning: a study of the influence of language upon thought and of the science of symbolism*, International library of psychology, philosophy and scientific method, Harcourt Brace Jovanovich, 1989, Online: https://books.google.de/books?id=M6UQAQAAIAAJ.

Charles Egerton Osgood, George J. Suci, and Percy H. Tannenbaum: *The Measurement of Meaning*, Illini books, University of Illinois Press, 1957, Online: http://www.press.uillinois.edu/books/catalog/32mtm7sx9780252745393.html.

Can Özmen, Alexander Streicher, and Andrea Zielinski: 'Using Text Segmentation Algorithms for the Automatic Generation of E-Learning Courses', in: *Proceedings of the Third Joint Conference on Lexical and Computational Semantics*, *SEM 2014, pp. 132–140, Dublin, Ireland, 2014, Online: http://www.aclweb.org/anthology/S14-1017.

Muntsa Padró, Marco Idiart, Aline Villavicencio, and Carlos Ramisch: 'Nothing like Good Old Frequency: Studying Context Filters for Distributional Thesauri', in: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, EMNLP 2014, pp. 419–424, Doha, Qatar, 2014, Online: http://www.aclweb.org/anthology/D14-1047.

Frank Robert Palmer: *Semantics*, Cambridge low priced editions, Cambridge University Press, 1981, Online: http://www.cse.iitk.ac.in/users/amit/books/palmer-1981-semantics.html.

Patrick Pantel, Eric Crestan, Arkady Borkovsky, Ana-Maria Popescu, and Vishnu Vyas: 'Web-scale distributional similarity and entity set expansion', in: *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, EMNLP 2009, pp. 938–947, Singapore, 2009, Online: https://aclweb.org/anthology/D/D09/D09-1098.pdf.

Patrick Pantel and Dekang Lin: 'Discovering word senses from text', in: *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD 2002, pp. 613–619, Edmonton, Alberta, Canada, 2002, Online: http://doi.acm.org/10.1145/775047.775138.

Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda: *English Gigaword Fifth Edition*, Linguistic Data Consortium, Philadelphia, 2011, Online: https://catalog.ldc.upenn.edu/LDC2011T07.

Pavel Pecina: 'Lexical association measures and collocation extraction', *Language Resources and Evaluation* 44: 137–158, 2010, Online: http://link.springer.com/article/10.1007%2Fs10579-009-9101-4.

Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi: 'WordNet::Similarity: measuring the relatedness of concepts', in: *Demonstration Papers at HLT-NAACL 2004*, pp. 38–41, Boston, MA, USA, 2004, Online: http://aclweb.org/anthology/N/N04/N04-3008.pdf.

Lev Pevzner and Marti A. Hearst: 'A Critique and Improvement of an Evaluation Metric for Text Segmentation', *Computational Linguistics* 28 (1): 19–36, 2002, Online: https://aclweb.org/anthology/J/J02/J02-1002.pdf.

Judita Preiss and Mark Stevenson: 'Unsupervised Domain Tuning to Improve Word Sense Disambiguation', in: *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL 2013, pp. 680–684, Atlanta, GA, USA, 2013, Online: http://www.aclweb.org/anthology/N13-1079.

William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery: *Numerical Recipes 3rd Edition: The Art of Scientific Computing*, Cambridge University Press, New York, NY, USA, 2007, Online: http://numerical.recipes/.

Carlos Ramisch: *A generic and open framework for multiword expressions treatment: from acquisition to applications*, Ph.D. thesis, Universidade Federal Do Rio Grande do Sul, 2012, Online: https://www.lume.ufrgs.br/bitstream/handle/10183/65777/000870122.pdf.

Carlos Ramisch, Vitor De Araujo, and Aline Villavicencio: 'A broad evaluation of techniques for automatic acquisition of multiword expressions', in: *Proceedings of the Student Research Workshop of the 50th Meeting of the Association for Computational Linguistics*, ACL 2012, pp. 1–6, Jeju Island, Republic of Korea, 2012, Online: http://www.aclweb.org/anthology/W12-3301.pdf.

Delip Rao, Paul McNamee, and Mark Dredze: 'Entity Linking: Finding Extracted Entities in a Knowledge Base', in: *Multi-source, Multilingual Information Extraction and Summarization*, Theory and Applications of Natural Language Processing, pp. 93–115, Springer Berlin Heidelberg, 2013, Online: http://dx.doi.org/10.1007/978-3-642-28569-1_5.

Reinhard Rapp: 'Mining Text for Word Senses Using Independent Component Analysis', in: *Proceedings of the 2004 SIAM International Conference on Data Mining*, pp. 422–426, Lake Buena Vista, FL, USA, 2004, Online: http://epubs.siam.org/doi/abs/10.1137/1.9781611972740.39.

Adwait Ratnaparkhi: 'A Maximum Entropy Model for Part-Of-Speech Tagging', in: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP 1996, pp. 133–142, Philadelphia, PA, USA, 1996, Online: http://www.aclweb.org/anthology/W96-0213.

Matthias Richter, Uwe Quasthoff, Erla Hallsteinsdóttir, and Chris Biemann: 'Exploiting the Leipzig Corpora Collection', in: *Proceedings of the Fifth Slovenian and First International Language Technologies Conference*, IS-LTC '06, pp. 68–73, Ljubljana, Slovenia, 2006, Online: http://wortschatz.uni-leipzig.de/~cbiemann/pub/2006/RichterQuasthoffHallsteinsdottirBiemann-ISLTC.pdf.

Martin Riedl, Irina Alles, and Chris Biemann: 'Combining Supervised and Unsupervised Parsing for Distributional Similarity', in: *Proceedings of the 25th International Conference on Computational Linguistics: Technical Papers*, COLING 2014, pp. 1435–1446, Dublin, Ireland, 2014a, Online: http://www.aclweb.org/anthology/C14-1136.

Martin Riedl and Chris Biemann: 'How Text Segmentation Algorithms Gain from Topic Models', in: *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-HLT 2012, pp. 553–557, Montreal, Canada, 2012b, Online: http://www.aclweb.org/anthology/N12-1064.

Martin Riedl and Chris Biemann: 'Sweeping through the Topic Space: Bad luck? Roll again!', in: *In Proceedings of the Joint Workshop on Unsupervised and Semi-Supervised Learning in NLP held in conjunction with EACL 2012*, ROBUS-UNSUP 2012, pp. 19–27, Avignon, France, 2012c, Online: http://www.aclweb.org/anthology/W12-0703.pdf.

Martin Riedl and Chris Biemann: 'TopicTiling: A Text Segmentation Algorithm based on LDA', in: *Proceedings of the Student Research Workshop of the 50th Meeting of the Association for Computational Linguistics*, pp. 37–42, Jeju, Republic of Korea, 2012a, Online: www.jlcl.org/2012_Heft1/jlcl2012-1-3.pdf.

Martin Riedl and Chris Biemann: 'From Global to Local Similarities: A Graph-Based Contextualization Method using Distributional Thesauri', in: *Proceedings of TextGraphs-8 Graph-based Methods for Natural Language Processing held in conjunction with EMNLP*, pp. 39–43, Seattle, WA, USA, 2013b, Online: http://aclweb.org/anthology/W13-5006.

Martin Riedl and Chris Biemann: 'Scaling to Large^3 Data: An efficient and effective method to compute Distributional Thesauri', in: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, EMNLP 2013, pp. 884–890, Seattle, WA, USA, 2013a, Online: https://aclweb.org/anthology/D/D13/D13-1089.pdf.

Martin Riedl and Chris Biemann: 'A Single Word is not Enough: Ranking Multiword Expressions Using Distributional Semantics', in: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP 2015, pp. 2430–2440, Lisboa, Portugal, 2015, Online: https://aclweb.org/anthology/D/D15/D15-1290.pdf.

Martin Riedl, Michael Glass, and Alfio Gliozzo: 'Lexical Substitution for the Medical Domain', in: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, EMNLP 2014, pp. 610–614, Doha, Qatar, 2014b, Online: http://emnlp2014.org/papers/pdf/EMNLP2014066.pdf.

Martin Riedl, Richard Steuer, and Chris Biemann: 'Distributed Distributional Similarities of Google Books over the Centuries', in: *International Conference on Language Resources and Evaluation*, LREC 2014, pp. 1401–1405, Reykjavik, Iceland, 2014c, Online: http://www.lrec-conf.org/proceedings/lrec2014/pdf/274_Paper.pdf.

Alan Ritter, Mausam, and Oren Etzioni: 'A Latent Dirichlet Allocation Method for Selectional Preferences', in: *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL 2010, pp. 424–434, Uppsala, Sweden, 2010, Online: http://www.aclweb.org/anthology/P10-1044.

Stephen Roller and Sabine Schulte im Walde: 'A Multimodal LDA Model integrating Textual, Cognitive and Visual Modalities', in: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1146–1157, Seattle, WA, USA, 2013, Online: http://aclweb.org/anthology/D13-1115.

Kevin Dela Rosa and Maxine Eskenazi: 'Effect of Word Complexity on L2 Vocabulary Learning', in: *Proceedings of the 6th Workshop on Innovative Use of NLP for Building Educational*

Applications in con junction with ACL-HLT 2011, BEA 2011, pp. 76–80, Portland, OR, USA, 2011, Online: http://www.aclweb.org/anthology/W11-1409.

Herbert Rubenstein and John B. Goodenough: 'Contextual correlates of synonymy', *Communications of the ACM* 8 (10): 627–633, October 1965, Online: http://doi.acm.org/10.1145/365628.365657.

Gerda Ruge: 'Experiments on linguistically-based term associations', *Information Processing & Management* 28 (3): 317 – 332, 1992, Online: http://www.sciencedirect.com/science/article/pii/030645739290078E.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams: 'Learning Representations by Back-propagating Errors', in: *Neurocomputing: Foundations of Research*, pp. 696–699, MIT Press, Cambridge, MA, USA, 1988, Online: http://dl.acm.org/citation.cfm?id=65669.104451.

Eugen Ruppert, Manuel Kaufmann, Martin Riedl, and Chris Biemann: 'JoBimViz: A Web-based Visualization for Graph-based Distributional Semantic Models', in: *Proceedings of the Annual Meeting of the Association for Computational Linguistics System Demonstrations*, ACL 2015, pp. 103–108, 2015a, Online: https://aclweb.org/anthology/P/P15/P15-4018.pdf.

Eugen Ruppert, Jonas Klesy, Martin Riedl, and Chris Biemann: 'Rule-based Dependency Parse Collapsing and Propagation for German and English', in: *Proceedings of the International Conference of the German Society for Computational Linguistics and Language Technology*, GSCL 2015, pp. 58–66, Duisburg, Germany, 2015b, Online: https://www.informatik.tu-darmstadt.de/fileadmin/user_upload/Group_LangTech/publications/RuppertEtAl_GSCL2015_collapsed_dependencies.pdf.

Ivan A. Sag, Timothy Baldwin, Francis Bond, Ann Copestake, and Dan Flickinger: 'Multiword Expressions: A Pain in the Neck for NLP', in: *Proceddings of the 3rd International Conference on Intelligent Text Processing and Computational Linguistics*, CICLing 2002, pp. 1–15, Mexico City, Mexico, 2001, Online: http://link.springer.com/chapter/10.1007%2F3-540-45715-1_1.

Magnus Sahlgren: *The Word-Space Model: using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces.*, Ph.D. thesis, Stockholm University, 2006, Online: eprints.sics.se/437/1/TheWordSpaceModel.pdf.

Bahar Salehi, Paul Cook, and Timothy Baldwin: 'Using Distributional Similarity of Multi-way Translations to Predict Multiword Expression Compositionality', in: *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, EACL 2014, pp. 472–481, Gothenburg, Sweden, 2014, Online: http://aclweb.org/anthology/E14-1050.

Gerard Salton, Andrew Wong, and Chung-Shu Yang: 'A Vector Space Model for Automatic Indexing', *Communications of the ACM* 18 (11): 613–620, November 1975, Online: http://doi.acm.org/10.1145/361219.361220.

Issei Sato and Hiroshi Nakagawa: 'Topic Models with Power-law Using Pitman-Yor Process', in: *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD 2010, pp. 673–682, Washington, DC, USA, 2010, Online: http://doi.acm.org/10.1145/1835804.1835890.

Ferdinand de Saussure: *Course in general linguistics*, Language (Philosophical Library), Philosophical Library, 1959, Online: http://books.google.de/books?id=FSpZAAAAMAAJ.

Martin Scaiano and Diana Inkpen: 'Getting More from Segmentation Evaluation', in: *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT 2012, pp. 362–366, Montreal, Canada, 2012, Online: http://www.aclweb.org/anthology/N12-1038.pdf.

Helmut Schmid: 'Probabilistic Part-of-Speech Tagging Using Decision Trees', in: *International Conference on New Methods in Language Processing*, pp. 44–49, Manchester, UK, 1994, Online: http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/data/tree-tagger1.pdf.

Helmut Schmid: 'Improvements In Part-of-Speech Tagging With an Application To German', in: *Proceedings of the ACL SIGDAT-Workshop*, pp. 47–50, Cambridge, MA, USA, 1995, Online: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.52.2255.

Patrick Schone and Daniel Jurafsky: 'Is Knowledge-Free Induction of Multiword Unit Dictionary Headwords a Solved Problem?', in: *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, EMNLP 2001, Pittsburgh, PA, USA, 2001, Online: http://aclweb.org/anthology/W01-0513.

Hinrich Schütze: 'Word Space', in: *Advances in Neural Information Processing Systems 5*, NIPS 5, pp. 895–902, Denver, CO, USA, 1993, Online: http://papers.nips.cc/paper/603-word-space.

Roy Schwartz, Omri Abend, Roi Reichart, and Ari Rappoport: 'Neutralizing linguistically problematic annotations in unsupervised dependency parsing evaluation', in: *Proceedings of the 49nd Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, ACL 2011, pp. 663–672, Portland, OR, USA, 2011, Online: https://aclweb.org/anthology/P/P11/P11-1067.pdf.

Djamé Seddah, Marie Candito, Benoit Crabbé, and Enrique Henestroza Anguiano: 'Ubiquitous Usage of a Broad Coverage French Corpus: Processing the Est Republicain corpus', in: *Proceedings of the Eight International Conference on Language Resources and Evaluation*, LREC

2012, pp. 3249–3254, Istanbul, Turkey, 2012, Online: http://www.lrec-conf.org/proceedings/lrec2012/pdf/1130_Paper.pdf.

Djamé Seddah, Reut Tsarfaty, Sandra Kübler, Marie Candito, Jinho D. Choi, Richárd Farkas, Jennifer Foster, Iakes Goenaga, Koldo Gojenola Galletebeitia, Yoav Goldberg, Spence Green, Nizar Habash, Marco Kuhlmann, Wolfgang Maier, Joakim Nivre, Adam Przepiórkowski, Ryan Roth, Wolfgang Seeker, Yannick Versley, Veronika Vincze, Marcin Woliński, Alina Wróblewska, and Eric Villemonte de la Clergerie: 'Overview of the SPMRL 2013 Shared Task: A Cross-Framework Evaluation of Parsing Morphologically Rich Languages', in: *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*, pp. 146–182, Seattle, WA, USA, 2013, Online: http://www.aclweb.org/anthology/W13-4917.

Yoav Seginer: 'Fast unsupervised incremental parsing', in: *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, ACL 2007, pp. 384–391, Prague, Czech Republic, 2007, Online: https://aclweb.org/anthology/P/P07/P07-1049.pdf.

Anders Søgaard: 'Semi-supervised condensed nearest neighbor for part-of-speech tagging', in: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, ACL-HLT 2011, pp. 48–52, Portland, OR, USA, 2011, Online: http://www.aclweb.org/anthology/P11-2009.pdf.

Anders Søgaard: 'Unsupervised dependency parsing without training', *Natural Language Engineering* 18 (02): 187–203, 2012, Online: http://journals.cambridge.org/action/displayAbstract?fromPage=online&aid=8511899&fileId=S1351324912000022.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii: 'BRAT: A Web-based Tool for NLP-assisted Text Annotation', in: *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, EACL 2012, pp. 102–107, Avignon, France, 2012, Online: http://www.aclweb.org/anthology/E12-2021.pdf.

Alexander Strehl, Er Strehl, Joydeep Ghosh, and Raymond Mooney: 'Impact of Similarity Measures on Web-page Clustering', in: *In Workshop on Artificial Intelligence for Web Search*, AAAI Workshop 2000, pp. 58–64, Austin, TX, USA, 2000, Online: https://www.aaai.org/Papers/Workshops/2000/WS-00-01/WS00-01-011.pdf.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum: 'Yago: A Core of Semantic Knowledge Unifying WordNet and Wikipedia', in: *Proceedings of the 16th International Conference on World Wide Web*, WWW 2007, pp. 697–706, Banff, Alberta, Canada, 2007, Online: www2007.org/papers/paper391.pdf.

Qi Sun, Runxin Li, Dingsheng Luo, and Xihong Wu: 'Text segmentation with LDA-based Fisher kernel', in: *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies*, ACL 2008, pp. 269–272, Columbus, OH, USA, 2008, Online: http://www.aclweb.org/anthology/P08-2068.

György Szarvas, Chris Biemann, and Iryna Gurevych: 'Supervised All-Words Lexical Substitution using Delexicalized Features', in: *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-HLT 2013, pp. 1131–1141, Atlanta, GA, USA, 2013, Online: https://aclweb.org/anthology/N/N13/N13-1133.pdf.

Stefan Thater, Georgiana Dinu, and Manfred Pinkal: 'Ranking Paraphrases in Context', in: *Proceedings of the 2009 Workshop on Applied Textual Inference in conjunction with the ACL '09*, TextInfer 2009, pp. 44–47, Suntec, Singapore, 2009, Online: http://www.aclweb.org/anthology/W09-2506.pdf.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer: 'Feature-rich Part-of-speech Tagging with a Cyclic Dependency Network', in: *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, NAACL 2003, pp. 173–180, Edmonton, Canada, 2003, Online: aclasb.dfki.de/nlp/bib/N03-1033.

Grigorios Tsoumakas and Ioannis Katakis: 'Multi-label classification: An overview', *International Journal of Data Warehousing and Mining* 2007: 1–13, 2007, Online: www.lpis.csd.auth.gr/publications/tsoumakas-ijdwm.pdf.

Masao Utiyama and Hitoshi Isahara: 'A statistical model for domain-independent text segmentation', in: *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, ACL 2001, pp. 499–506, Toulouse, France, 2001, Online: https://aclweb.org/anthology/P/P01/P01-1064.pdf.

Gabriella Vigliocco, Pamela Perniss, and David Vinson: 'Language as a multimodal phenomenon: implications for language learning, processing and evolution', *Philosophical Transactions of the Royal Society of London B: Biological Sciences* 369 (1651): 1–7, 2014, Online: http://rstb.royalsocietypublishing.org/content/369/1651/20130292.

Hanna Wallach, David Mimno, and Andrew McCallum: 'Rethinking LDA: Why priors matter', in: *Advances in Neural Information Processing Systems*, pp. 1973–1981, Vancouver, B.C., Canada, 2009, Online: http://papers.nips.cc/paper/3854-rethinking-lda-why-priors-matter.

Charles Wayne: 'Topic Detection and Tracking (TDT): Overview & Perspective', in: *Proceedings of the Broadcast News Transcription and Understanding Workshop*, Lansdowne, VA, USA, 1998, Online: http://lpis.csd.auth.gr/publications/tsoumakas-ijdwm.pdf.

Warren Weaver: 'Translation', in: *Machine Translation of Languages*, pp. 15–23, MIT Press, Cambridge, MA, 1949/1955, Online: www.mt-archive.info/Weaver-1949.pdf. Reprinted from a memorandum written by Weaver in 1949.

Julie Weeds: *Measures and Applications of Lexical Distributional Similarity*, Ph.D. thesis, Department of Informatics, University of Sussex, England, 2003, Online: http://citeseer.ist.psu.edu/weeds03measures.html.

Julie Weeds, David Weir, and Diana McCarthy: 'Characterising measures of lexical distributional similarity', in: *Proceedings of the 20th international conference on Computational Linguistics*, COLING 2004, pp. 1015–1021, Geneva, Switzerland, 2004, Online: www.aclweb.org/anthology/C04-1146.

Georg Wells: *The Meaning Makers: Children Learning Language and Using Language to Learn*, Hodder and Stoughton, 1987, Online: http://eric.ed.gov/?id=ED264572.

Joachim Wermter and Udo Hahn: 'Effective grading of termhood in biomedical literature', in: *Annual AMIA Symposium Proceedings*, pp. 809–813, Washington D.C., USA, 2005, Online: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1560898/.

Terry Winograd: *Understanding Natural Language*, Academic Press, Inc., 1972, Online: http://dl.acm.org/citation.cfm?id=540414.

Ludwig Josef Johann Wittgenstein: *Philosophische Untersuchungen*, Suhrkamp Verlag, 1953.

Yaakov Yaari: 'Segmentation of Expository Texts by Hierarchical Agglomerative Clustering', in: *Proceedings of the Conference on Recent Advances in Natural Language Processing*, RANLP 1997, pp. 59–65, Tzigov Chark, Bulgaria, 1997, Online: http://arxiv.org/abs/cmp-lg/9709015.

Xuchen Yao and Benjamin Van Durme: 'Nonparametric Bayesian Word Sense Induction', in: *Proceedings of TextGraphs-6: Graph-based Methods for Natural Language Processing held in conjunction with ACL-HLT '11*, TextGraphs-6, pp. 10–14, Portland, OR, USA, 2011, Online: https://aclweb.org/anthology/W/W11/W11-1102.pdf.

Menno van Zaanen: 'Building Treebanks Using a Grammar Induction System', *Technical report*, University of Leeds, School of Computer Studies, 2001, Online: http://books.google.de/books?id=i5oXMwEACAAJ.

Jiajun Zhang, Feifei Zhai, and Chengqing Zong: 'Handling Unknown Words in Statistical Machine Translation from a New Perspective', in: *Natural Language Processing and Chinese Computing*, Communications in Computer and Information Science Vol. 333, pp. 176–187, Springer Berlin Heidelberg, 2012, Online: http://link.springer.com/chapter/10.1007%2F978-3-642-34456-5_17.

George Kingsley Zipf: 'Relative frequency as a determinant of phonetic change', *Reprinted from the Harvard Studies in Classical Philology* 1929, Online: www.jstor.org/stable/310585.

Index